For Macintosh Programmers & Developers

MacTech™ MAGAZINE

Vol. 12, No. 5 • May 1996

Stitching Into Virtual Reality!

INSIDE:

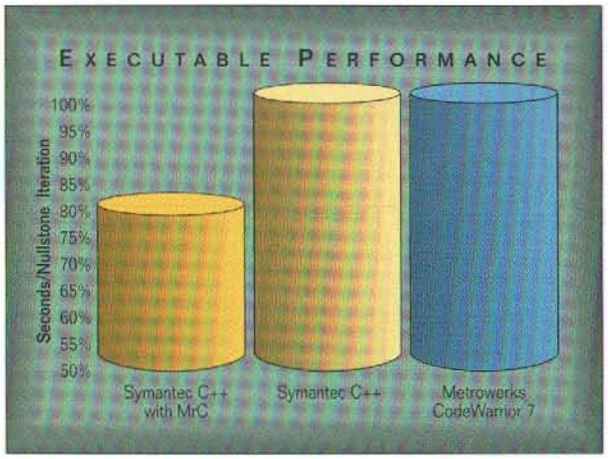
$5.85 US
$6.95 Canada
ISSN 1067-8360
Printed in U.S.A.

05
0 32128 74887 8

Now, through a joint development agreement, Symantec and Apple™ Computer let you produce the fastest Power Mac code.

Symantec C++ for Power Macintosh™ now comes with MrC– Apple's new optimizing compiler.

Industry-standard Nullstone tests show that



*Code compiled using MrC runs an average of 22% faster than code compiled with the standard Symantec C++ compiler or Metrowerks Code Warrior 7™.**

size and color. Organizing and navigating a project has never been easier.

In addition, Symantec C++'s multi-threaded environment gives you the ability to edit and write code while you compile. And our visual architect lets you quickly draw the interface. Corresponding code is then generated automatically.

WHEN MRC GOES TO WORK, SYMANTEC C++ APPLICATIONS GET 22% FASTER.

applications compiled with MrC run an average of 22% faster.

DEVELOP FASTER APPLICATIONS FASTER.

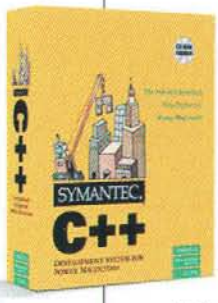
Not only can you develop the fastest Power Mac applications, you can write them fast, too. New AppleScript support lets you automate repetitive tasks. While the new linker provides fast turnaround for incremental builds.

With the fully integrated class browser, you'll quickly navigate your C++ class library. And support for templates and multiple inheritance further boosts your productivity.

ORGANIZE AND MANAGE PROJECTS EASILY.

The new Project Manager lets you organize and manage nested projects. You can display hierarchical groups as folders within a project window for easier organization.

The editor gives you unlimited split panes and full text formatting– font,



MRC COMPILER
produces the fastest Power Mac applications

APPLESCRIPT
automates the build process

NEW INCREMENTAL LINKER
provides fast incremental builds

TEMPLATE AND MULTIPLE INHERITANCE SUPPORT
increases productivity

NESTED PROJECTS AND FOLDERS
let you organize and navigate projects

MULTI-THREADED ENVIRONMENT
lets you edit and write code while compiling

VISUAL ARCHITECT
builds your interface visually

All in all, Symantec C++ is a great way to develop the fastest applications for Power Mac.

THREE CDS FOR THE PRICE OF ONE.

When you register as a Symantec C++ owner, you'll be enrolled in the Symantec C++ Subscription for Macintosh program. Subscribers will automatically receive two free product updates (on CD ROM) so you'll always have the latest features and tools.

Learn more about Symantec C++ on the Internet at www.symantec.com Or call 1-800-628-4777, Extension 9H22 for more information.

SYMANTEC.®

Eddy Award Winner** for **Best New Developer Tool
– MacUser Editors Choice Awards, 1993

"A distinct improvement over ResEdit."
– MacTech / MacTutor

"Resorcerer's data template system is amazing!"
– Bill Goodman, author of Compact Pro

"Nuke ResEdit! Resorcerer is mission-critical for us."
– Dave Winer, Userland Frontier

"The color pixel editors are wonderful! A work of art!"
– Dave Winzler, author of Microseeds Redux

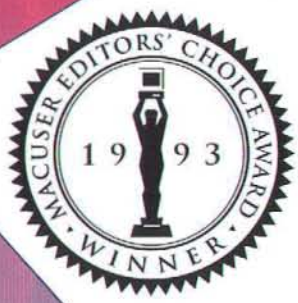
"Every Macintosh developer should own a copy of Resorcerer."
– Leonard Rosenthol, Aladdin Systems

"Resorcerer will pay for itself many times over in saved time and effort."
– MacUser review

"The template that disassembles 'PICT's is awesome!"
– Bill Steinberg, author of Pyro! and PBTools

"Resorcerer proved indispensible in its own creation!"
– Doug McKenna, author of Resorcerer

"...a wealth of time-saving tools."
MacUser Review, Dec. 1992



RESORCERER®

Version 1.2.4

The Resource Editor for the Macintosh Wizard

ORDERING INFO

Needs: ≥Mac Plus, ≥ Sys 4.2, 1MB
Likes: ≥Mac Plus, ≥ Sys 7.0, 2MB
32-bit clean, AU/X compatible

Price: $256 (decimal)
(Educational, quantity, or other discounts available)

Includes: 500 page manual
60-day Money-Back Guarantee
Domestic UPS ground shipping

Payment: Check, PO's, or Visa/MC

Extras (call us):
COD, FedEx, UPS Blue/Red, International Shipping

Downloadable Demos/Updaters:
AppleLink: Software Sampler
AOL: Software Libs/Development
CompuServe: MACDEV/Tools
or call us.

New 1.2 Features:

- **New 'cicn', 'ppat', 'crsr', 'acur', 'pltt', 'clut' editors**
- **Powerful icon family editing (all 9 icon types)**
- **Color pixel anti-aliasing, dithering, and lots more**
- **Complete 'PICT' disassembly and reassembly**
- **Resource sorting; ROM resource browsing**
- **120 template field parsing types now supported**
- **New insertion & deletion template field types**
- **Text-only 'PICT' resources**
- **Lots of improvements throughout**

- Easier, faster, more Mac-like, and more productive than ResEdit
- Safer memory-based, not disk-file-based, design and operation
- All file information and common commands in one easy-to-use window
- Compares resource files, and even **edits your data forks** as well
- Visible, accumulating, editable scrap
- Searches and opens/marks/selects resources by text content
- Makes global resource ID or type changes easily and safely
- Builds resource files from simple Rez-like scripts
- Most editors DeRez directly to the clipboard
- All graphic editors support screen-copying or partial screen-copying
- Hot-linking Value Converter for editing 32 bits in a dozen formats
- Its own 32-bit List Mgr can open and edit very large data structures
- Templates can pre- and post-process any arbitrary data structure
- Includes nearly 200 templates for common system resources
- TMPLs for Installer, MacApp, QT, Help, AppleEvent, OCE, GX, etc.
- Full integrated support for editing color dialogs and menus
- Try out balloons, 'ictb's, lists and popups, even create C source code
- Integrated single-window Hex/Code Editor, with patching, searching
- Editors for cursors, versions, pictures, bundles, and lots more
- Well-designed, helpful developer tools being added all the time
- Relied on by thousands of Macintosh developers around the world

MATHEMÆSTHETICS, INC.
P.O. Box 298 • Boulder • CO • 80306-0298 • USA
Phone: (303) 440-0707 • Fax: (303) 440-0504
AppleLink/AmericaOnline: RESORCERER • Internet: resorcerer@aol.com

THE STAFF

For Macintosh Programmers & Developers

MacTech™ MAGAZINE

A publication of XPLAIN CORPORATION

How To Communicate With Us

In this electronic age, the art of communication has become both easier and more complicated. Is it any surprise that we prefer **e-mail?** If you have any questions, feel free to call us at 805/494-9797 or fax us at 805/494-9798.

DEPARTMENTS	Internet	CompuServe	AppleLink
Orders, Circulation, & Customer Service	custservice@mactech.com	71333,1063	–
Press Releases	press_releases@mactech.com	–	–
Ad Sales	adsales@mactech.com	–	–
Editorial	editorial@mactech.com	71333,1065	–
Programmer's Challenge	progchallenge@mactech.com	–	–
Online Support	online@mactech.com	–	–
Accounting	accounting@mactech.com	–	–
Marketing	marketing@mactech.com	–	–
General	info@mactech.com	71333,1064	MacTechMag
Online support area	http://www.mactech.com		

MACTECH MAGAZINE

MacTech Magazine is grateful to the following individuals who contribute on a regular basis. We encourage others to share the technology.

We are dedicated to the distribution of useful programming information without regard to Apple's developer status.

*For information on submitting articles, ask us for our **writer's kit** which includes the terms and conditions upon which we publish articles.*

Editors

Publisher/Editor-in-Chief • Neil Ticktin
Managing Editor • Matt Neuburg
Editorial Assistant • John Kawakami
Editor Emeritus • Scott T Boyd
Editor-at-Large • Eric Gundrum

Contributing and Technical Editors

Copland • Steve Kiene, Mindvision
Internet • Jon Wiederspan
MagicCap/TeleScript • Richard Clark, General Magic
OpenDoc • Tantek Çelik
Performance Programming • Jim Gochee
Tips & Tidbits • Steve Sisak

Regular Columnists

Getting Started • Dave Mark
Programmer's Challenge • Bob Boonstra
Insider's View • Jordan Mattson
Symantec Top 10 • Symantec Technical Support
Crabb's Apple • Don Crabb
URLs • Jim Straus

XPLAIN CORPORATION

Chief Executive Officer • Neil Ticktin
Chief Operating Officer • Andrea J. Sniderman
Controller • Brian Shin
Advertising Executive • Ruth Subrin
Art Director • Judith Chaplin
Customer Service • Al Estrada
Vendor Relations • Duk Im
Administrative Assistant • Susan Pomrantz
Network Administrator • Donal Corcoran
Software Engineer • Don Bresee

Board of Advisors • Steven Geller, Jordan Mattson, Blake Park, and Alan Carsrud

PRINTED WITH SOY INK

This publication is printed on paper with recycled content.

MacTech Magazine (ISSN: 1067-8360 / USPS: 010-227) is published monthly by Xplain Corporation, 850-P Hampshire Road, Westlake Village, CA 91361-2800. Voice: 805/494-9797, FAX: 805/494-9798. Domestic subscription rates are $47.00 per year. Canadian subscriptions are $59.00 per year. All other international subscriptions are $97.00 per year. Domestic source code disk subscriptions are $77 per year. All international disk subscriptions are $97.00 a year. Please remit in U.S. funds only. Second Class postage is paid at Thousand Oaks, CA and at additional mailing office.

POSTMASTER: Send address changes to **MacTech Magazine**, P.O. Box 5200, Westlake Village, CA 91359-5200.

TABLE OF CONTENTS

May 1996 • Volume 12, No.5

For Macintosh Programmers & Developers

MacTech™ MAGAZINE

VIEWPOINT

By Scott T Boyd, Editor Emeritus, http://www.montara.com/

Way back in 1984 Apple did something fundamentally different with the Macintosh that few people talk about when singing Apple's praises for changing computing's paradigm.

We came to know it as the A-trap mechanism. For those who haven't delved into this before, it's a little oddity made possible by Motorola's chip designers. Many people might not have even noticed a feature which captured the imagination of a few special people at Apple. 68xxx processors have a number of instructions or conditions which generate exceptions. Each of these causes an exception handler to get called. One entire class of exceptions is triggered by any instruction whose hex value begins with an $A (e.g. $ABCD, $AAFA, etc...). Those wise guys at Apple figured out that they could have the A-line exception code use the instruction's value as something of an index to figure out which function to further dispatch to. This we now know as the A-trap dispatcher.

The greatest disadvantage of this approach was execution time. It takes time to "throw" an exception. The processor has to put a fair amount of its state onto the stack before it can jump to the exception vector (and restore it later). Then the dispatcher has to do some fancy footwork to figure out who to call, and sometimes it saves and restores even more registers.

Now for the upside. First, a two byte trap word is smaller than any jump you could hope to make into a library of functions. On a 128K machine, that kind of savings really added up. Second, it made it easy to conceive of a vast library of functions that any code could call. While A-traps weren't necessary for things like the Macintosh Toolbox, they sure didn't hurt. They provided a consistent mechanism to dispatch to code anywhere on the machine, be it in the ROM, in RAM, or even on the disk, using a standard lookup key (the trap word). Third, it created a mechanism whereby Apple engineers and third parties could correct, modify, and extend the vast library of code (we know this as patching and/or implementing traps).

For years Macintosh programmers felt special. A-traps offered a rich breeding ground for unusual software. We customized our machines in ways not even imaginable on other (lesser?) machines. We had more fun and productivity to show for it, as well as a thriving utility software market.

Alas, the A-trap mechanism might very well have been too useful. You see, there came a time with other operating systems started to face some of the problems that Macintosh had already confronted. Only by this time these machines were already an order of magnitude larger. Perhaps their OS engineers weren't as clever as Apple's, but they didn't seem particularly interested in paying the price of exception handling to do a late-binding method dispatch (which is, after all, essentially what we've been talking about). While Apple was busily guarding and carefully managing the growth of the trap tables, other OS vendors were coming up with these things called DLLs – Dynamically Linked Libraries.

DLLs are like object code libraries, but aren't linked directly into an application, and can generally be shared. In that sense, they can become like system software, with just one copy of a library per machine. If you put some system functionality into a DLL, you can often ship it by itself rather than wait for an entire system release. Apple has long achieved this effect by shipping extensions which, when installed, add new functionality at boot time. DLLs have the benefit, though, of not needing to load at boot time, and not having to live in a special place on disk. DLLs also have the benefit of freedom from the space constraints that trap code typically tries to live by. With the addition of file mapping and a good virtual memory system, they can even execute directly off of disk. Have a good night programming and speed up a DLL by 50%? Ship the new DLL and watch all of the apps which use it simply go faster! Other operating systems mastered this before Apple even realized that having a DLL mechanism would be a good thing to have.

Apple dabbled with ASLM, the Component Manager, and others. Fortunately, the PowerPC effort brought forth the Code Fragment Manager, the one we've been waiting for, and now 68K CFM is ready, too. I've recently been involved in an effort to build some fat code fragments for use by a dynamic language environment, and I've got to tell you that I am *pumped!*

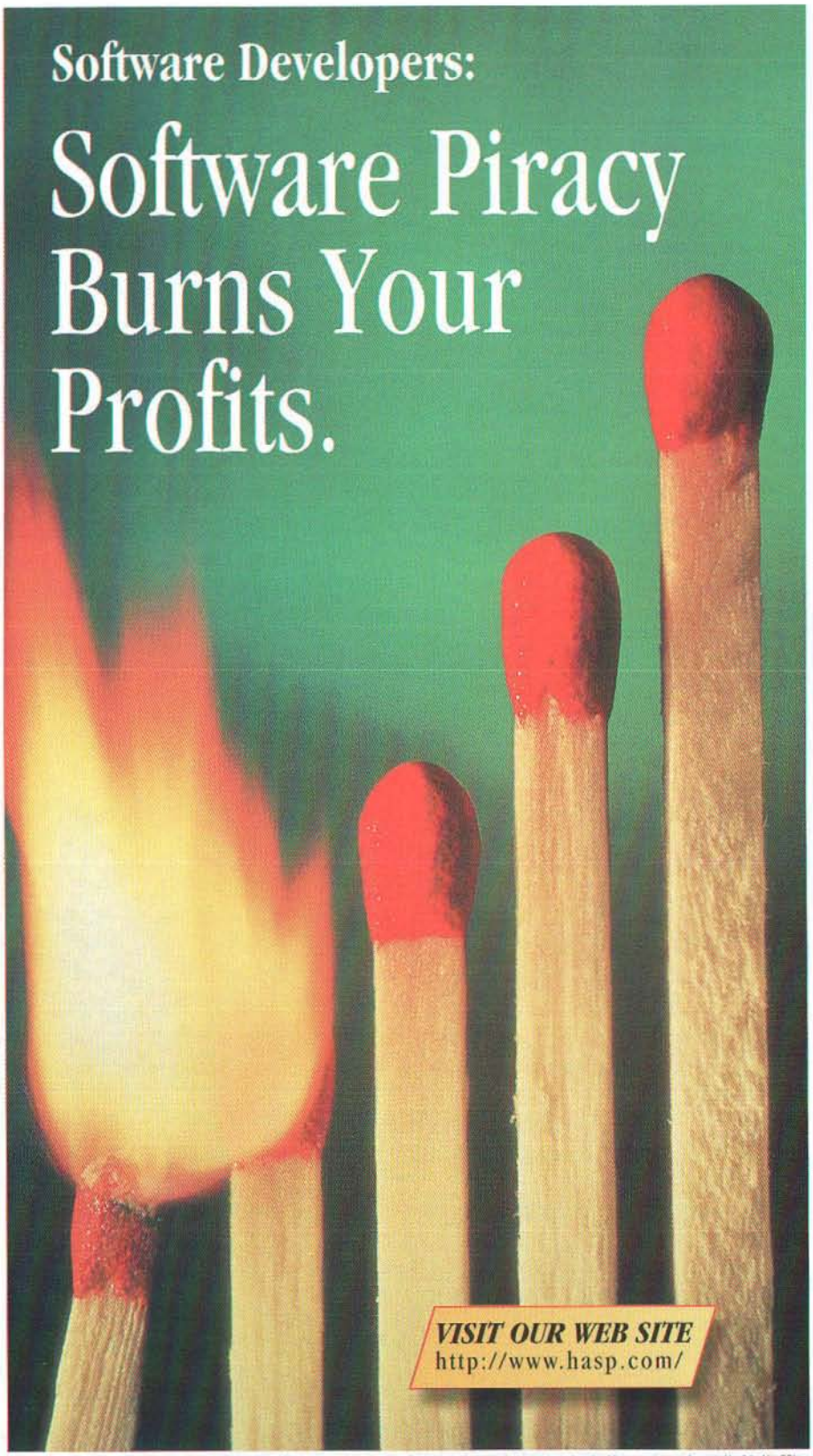
But now I find that I want lots more fragments to play with. Wouldn't you like to have a DLL that converted PICTs to GIF, told your Web browser to resolve a URL, or went a ftp'd a file, all with the ease of a couple of function calls? Here's where you come in. You know that really clever or useful library you wrote and linked in to your latest application? How about that slick little shareware utility? How about releasing it as a DLL? That and a header file, and you can have thousands of grateful developers treat you like their own system software engineer. CFM is cool. If you learn just one new thing today, learn what CFM can do for you. One starting point is :
http://dev.info.apple.com/evangelism/DLLDirectional.html
I also recommend looking at Apple's latest Developer CDs.

Food for Thought

Presenting his keynote to the Microsoft Professional Developers Conference, Bill Gates noted that he believes that the #1 reason to develop for Windows is "Volume!" That's 60 million copies of Windows each year. As an aside, he mentioned, "30 million copies wouldn't pay Microsoft's bills."

Software Developers:

Software Piracy Burns Your Profits.



Each year, the illegal use of software consumes nearly 50% of your potential revenues. With the flames of piracy eating away at your profits, can you afford not to protect your software?

Software Obtained Illegally, by region, 1993 vs. 1994

MacHASP® is widely acclaimed as the world's most advanced software protection solution for Macintosh computers. Since 1984, thousands of leading Mac and PC developers have used over one million MacHASP and HASP keys to protect billions of dollars worth of software. Why? Because MacHASP's security, reliability, and ease-of-use led them to a simple conclusion: MacHASP is the most effective software protection system available.

Today, more developers are choosing MacHASP than any other software protection method. To learn why, and to see how easily you can increase your revenues, call now to order your MacHASP Developer's Kit.

1-800-223-4277

ALADDIN

The Professional's Choice

North America — **Aladdin Software Security Inc.**
Tel: (800) 223 4277, 212-564 5678
Fax: 212-564 3377
E-mail: sales@hasp.com
WWW: http://www.hasp.com/

Intl Office — **Aladdin Knowledge Systems Ltd.**
Tel: 972-3-537 5795, Fax: 972-3-537 5796
E-mail: aladdin@aladdin.co.il

United Kingdom — **Aladdin Knowledge Systems UK Ltd.**
Tel: 01753-622266, Fax: 01753-622262

France — **Aladdin France SA**
Tel: 1 40 85 98 85, Fax: 1 41 21 90 56

member of

yes It runs with NetWare

Available and Ready for OS/2

Microsoft Windows Compatible

Mac OS

Aladdin Benelux 08894 19777 ■ **Aladdin Japan** 0426 60 7191 ■ **Aladdin Russia** 095 9230588 ■ **Australia** Conlab 3 8985685 ■ **Czech** Atlas 2 766085 ■ **Chile** Micrologica 2 222 1388 ■ **Denmark** Berendsen 39 577300 ■ **Egypt** Zeineldein 2 3604632 ■ **Finland** ID-Systems 0 870 3520 ■ **Germany** CSS 201 278804 ■ **Greece** Unibrain 1 6856320 ■ **India** Solutions 11 2218254 ■ **Italy** Partner Data 2 26147380 ■ **Korea** Dae-A 2 848 4481 ■ **Mexico** SiSoft 5 5439770 ■ **New Zealand** Training 4 5666014 ■ **Poland** Systherm 61 480273 ■ **Portugal** Futurmatica 1 4116269 ■ **Romania** Interactiv 64 153112 ■ **South Africa** D.I.e Roux 11 886 4704 ■ **Spain** PC Hardware 3 4493193 ■ **Switzerland** Opag 61 7169222 ■ **Taiwan** Teco 2 555 9676 ■ **Turkey** Mikrotesta 312 467 7504

Power Too Abundant to Meter

neoAccess™
Cross-Platform Object Database Engine

Powerful
NeoAccess and NeoShare are the most powerful object-oriented database engines available. They display electrifying performance—up to ten times that of competitors. Behind an elegant programming interface is a high performance query engine utilizing: extended binary trees and binary search algorithms tuned for short access times, dynamically combined, collapsed, and compressed indices, and object caching for lightning fast access to previously used objects.

No Runtime Fees
Get the power of NeoAccess and avoid the expense and administrative hassle of feeding the runtime fees meter. You pay one affordable price no matter how many copies of your application you sell or use.

Cross Platform
Others may promise cross-platform development tools—NeoLogic delivers. NeoAccess and NeoShare consists of C++ classes designed for use with popular compilers and application frameworks on Windows® (3.x, '95 and NT), Macintosh® (M68k and PPC), and Unix™ (all flavors) platforms. Full source code is available so both products can even be used with custom frameworks.

New in NeoShare Version 2.0
Cross-platform and internet-savvy NeoShare 2.0 is now shipping! NeoShare extends standard application frameworks including: Metrowerks' PowerPlant, Symantec's THINK Class Library and Apple's MacApp on the Macintosh, and Microsoft's Foundation Classes and Borland's OWL in Intel-based environments.

neoShare™
Client/Server Object Database Engine

Scalable
NeoShare extends the core features of NeoAccess to client/server applications. NeoShare includes the NeoAccess Toolkit and everything you need to create data intensive distributed applications. Its advanced client/server architecture provides shared access to objects by multiple inter- and intra-networked clients. Combine client and server functions into a single application or create separate client and server applications. With NeoShare you have complete flexibility in the design of your distributed systems.

Proven
Thousands of commercial and in-house developers have already found that NeoLogic's technology enabled them to build fast, lightweight and powerful applications in record time. That's why NeoAccess and NeoShare based applications are already operating on millions of computers. Tap into the power for your next development project!

neo•logic™

Powering Development of Object-Oriented Applications

NeoLogic Systems 1450 Fourth St., Suite 12 v. 510.524.5897
neologic@neologic.com Berkeley, CA 94710 f. 510.524.4501

GETTING STARTED

By Dave Mark

Java Break

Over the next few months, we're going to take a break from PowerPlant and play around with the Java programming language. Why spend time on Java? Java is extremely popular. Java is a cool language. Many people think Java is a better language than both C and C++. Java offers an elegant mechanism for developing software that will run on a Mac, Windows machine, or a Unix box, all without recompiling.

If you are even a little interested in the World Wide Web, you should definitely learn Java. While you'll use HTML (or an HTML-generating program) to create your web content, you can greatly enhance your Web pages by calling up Java applets from within your HTML.

Java Is Not C++

You will frequently see Java compared to C++. Though Java is very similar to C++, there are many major differences. For starters, Java doesn't support pointers, is designed to support multi-threading, and does its own garbage collection and dynamic memory management. To add a user interface to a C++ program, you'll take advantage of a Toolbox designed for a specific platform (in our case, we use the Mac Toolbox). To add an interface to a Java applet, you'll take advantage of a Java-specific, platform-independent interface library called the AWT (advanced windowing toolkit) that ships with your development environment.

Java is, however, very similar to C++. As we go through our Java examples, you'll find very few clues that this isn't straight C++ code. Of course, I'll do my best to point out the differences.

Before you read on, you might want to take some time to get a bit of background on the Java language. The March issue of *MacTech Magazine* had an excellent article by Richard Cardona called *Writing Java 102* that is definitely worth a read. Of course, there have been a number of other Java articles, and there is also a steadily increasing deluge of Java books hitting the market. Check 'em out, pick one you like, dig in and learn the basics.

Getting Java on Your Machine

Before you can work with Java, you'll need to get hold of a Java development environment. There are several choices. For starters, you can visit Sun's Java web site (http://java.sun.com) and download the Macintosh version of the **Java Development Kit** (JDK). The JDK is free (except, of course, for any internet access fees) and fairly straightforward to use. It consists of all the files you need to do Java development, along with a Java compiler and an applet runner called Applet Viewer. The real problem with the JDK is that it is not a development environment, but rather a collection of tools. To compile a file, you either drop the file onto the compiler or, from inside the compiler, you select **Open...** from the **File** menu and select the file you want to compile. To edit your source code, you use SimpleText, BBEdit, or some other plain text editor. To run the compiled `.class` file, you go back to the Finder and drag the `.class` file onto the Applet Viewer. All this back-and-forth between tools and the Finder is a real pain. On the other hand, I don't think Sun really intended the JDK to be the programmer's choice for Java development.

Another choice for Java development is **Roaster**, which was released in January by Natural Intelligence. Roaster uses an integrated development environment (like CodeWarrior and SC++), allowing you to edit, compile, and run your applets from within the environment. The January release of Roaster sold for $299.

By the time you read this, Metrowerks will have shipped **Discover Java**, their professional Java development environment. Discover Java sells for $99, and comes with an electronic copy of *Learn Java on the Macintosh* by Barry Boone

(Addison-Wesley, 1996). The Java development tools also ship on CodeWarrior Gold 9, so if you already subscribe to CW Gold, about all you'll be missing is the book. Since I'm writing this about three months before Discover Java and CW9 hit the streets, check with Metrowerks (http://www.metrowerks.com, or call (800) 377-5416 or send email to sales@metrowerks.com) to get the latest on pricing and availability.

Finally, Symantec has announced Java support for the Macintosh via a product called **Café**. According to a press release on their Web site, Café will ship on the SC++ CD and was scheduled to ship in the first quarter of 1996.

Your First Application: Hello

Before we dig in to our first example, a quick word about terminology. Strictly speaking, the term **applet** applies to a Java class that is derived from the class `java.applet.Applet` and launched from an HTML page via the `<applet>` tag. Since none of our first few examples fit this definition, they are not really applets. On the other hand, these little examples aren't really applications in the true sense of the word, since that implies that they run independent of our Java environment. However, since the term "applet" is so specific, we'll stick with the term "application" until next month when we build our first true applet.

Our first Java application will display the obligatory "Hello, world!" message:

```
public class hello
{
  public static void main(String argv[])
  {
    System.out.println("Hello, world!");
  }
}
```

Create a new file, type in this code, then save the file as `hello.java`. Just as you'd end your C source code file with the extension `.c` and your C++ source code file with `.cp` or `.cpp`, the extension `.java` is used to denote a Java source code file.

If you are using the JDK, drag the file `hello.java` onto the application named Java Compiler (gee, guess what this is). The compiler will convert your source code into Java byte code, intended for a Java interpreter that will turn these byte codes into the instructions specific to the machine it is running on. The interpreter is part of the **virtual machine**, the layer that lies between the platform-independent Java byte code and your machine.

The compiler writes the Java byte code into a file named `xxx.class`, where `xxx` is the name of the class you've implemented. In this case, the compiler will create a file called `hello.class`. Again, if you are using the JDK, drag the file `hello.class` onto the application named Applet Viewer. As you might expect, the Applet Viewer will start running at the public method named `main()`.

The output of this application is shown in Figure 1. Notice that the window is named **stdout**. If you've ever spent any time in the Unix universe, you know that `stdout` stands for "standard output". The method `System.out.println()` sends its output to the **stdout** window followed by a carriage return. The method `System.out.print()` sends its output to the **stdout** window without generating a carriage return.

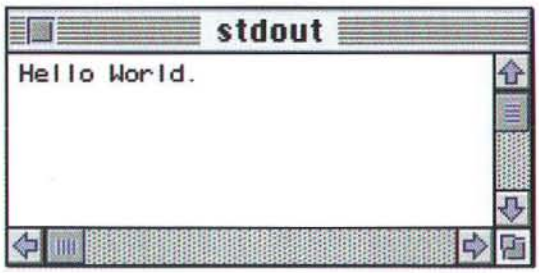


Figure 1. The output from the hello.class application

`System.out.println()` is a lot like `printf()` or `cout` in that they all send their output to a console window. Just as you first learned to program using consoles and eventually moved to the Mac Toolbox for your user interface, we'll start with `System.out.println()` and eventually move on to the user interface routines in the AWT (advanced windowing toolkit). You'll use the AWT to implement a user interface you'll want to appear on a web page.

Don't worry too much about the structure of our Java source code just yet. The three applications in this month's column all use the same basic structure: a class wrapper with a single public method called `main()`. Oh yeah, in Java, a class' functions are called methods instead of member functions.

Your Second Application: StringTester

Our second application, stringTester, introduces an important Java data type: `String`. As its name suggests, the `String` class was implemented to work with strings. Unlike C and Pascal strings, a Java `String` is an object, complete with variables (the Java term for data members) and methods. Our third application, stringMethods, will demonstrate some of the `String` methods. This application, stringTester, will get us started.

Here's the source...

```
public class stringTester
{
  public static void main(String argv[])
  {
    String string1, string2 = ", world!";

    string1 = "Hello";

    System.out.print( string1 );
    System.out.println( string2 );
    System.out.println( string1 + string2 );

    string1 += string2;
    System.out.println( string1 );

    System.out.println( "Length of this string: " +
              string1.length() );
  }
}
```

The first few lines show you two ways to create and initialize a `String`:

Announcing

MICROGUARD PLUS.™

Why so many developers are switching to MicroGuard copy protection

MicroGuard is committed to uncompromising technological superiority

"Technology at its peak" is our commitment to you. That is why we have brought you **MicroGuard Plus™**. And, MicroGuard Plus is **100% backwards** compatible with MicroGuard. This means MicroGuard and MicroGuard Plus can be used interchangeably. Just as we promised!

MicroGuard has now surpassed its own technological lead, actually improving on the best

MicroGuard Plus is everything MicroGuard is, plus 40-bit encryption, two additional passwords, 64-Bit Array, 32-byte public area, 45% size reduction, enhanced counter and more. In addition, MicroGuard Plus offers two new utilities: **QuickGuard™** and EasyGuard™ allow you to protect your applications without touching your source code. The only feature that is not plus is the price. :-)

MicroGuard offers you the most sophisticated network protection

Our network protection, **MicroGuard Net™**, is so superior, we had to hire an Apple network engineer to execute our specifications.

MicroGuard is the best selling Macintosh key in the world

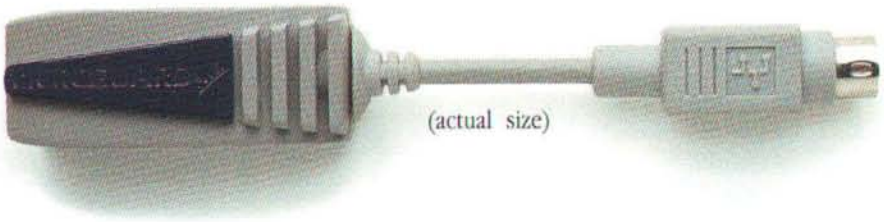
MicroGuard sells more Macintosh copy-protection keys than anyone else in the world!

MicroGuard is the only key developed by Mac developers, and is the first and only 100% ADB savvy key

We have been developing Mac applications as a seed development house since 1984. We are not a PC protection company that has come to you with a Mac product. MicroGuard is fully ADB savvy and offers extended addressing. Unlike other protection devices, MicroGuard never clashes with other keys. Only MicroGuard offers this level of sophistication.

MicroGuard delivers developer support within 24 hours

We will answer any inquiry you have within 24-hours. We also have a fully loaded AppleLink bulletin board which contains all our libraries, tech notes, Q&A and nearly everything you'll ever need!



For more information and to order a Developer's Kit or to receive a free CD ROM about MicroGuard, please contact us at:

MicroGuard USA: Tel: (303) 320-1628 • Fax: (303) 320-1599 • AppleLink: M.GUARD
International: Tel: (972) 3 558-2345 • Fax: (972) 3 558-2344 • AppleLink: MICROGUARD

Mac OS

Pirate Busters

```
String string1, string2 = ", world!";

string1 = "Hello";
```

You can initialize the `String` when you define it or you can use the assignment operator, as we did in the second line. Nothing unusual here.

The next line prints the first string, "Hello", without a carriage return. The second line prints the second string, ", world!", with a carriage return.

```
System.out.print( string1 );
System.out.println( string2 );
```

This produces this line of output in the **stdout** window:

```
Hello, world!
```

The next line uses the + operator to concatenate `string1` and `string2`, sending the joined string as a parameter to `println()`.

```
System.out.println( string1 + string2 );
```

Here's the output produced by this code:

```
Hello, world!
```

Next, the += operator is used to concatenate `string2` onto the end of `string1` and the new `string1` is send to **stdout**:

```
string1 += string2;
System.out.println( string1 );
```

Once again, here is the output:

```
Hello, world!
```

Finally, the `length()` method is called to return the length of the modified `string1`. Notice that the + operator is used to merge the two strings passed to `println()` into a single string:

```
System.out.println( "Length of this string: " +
        string1.length() );
```

Here's the final line of output:

```
Length of this string: 13
```

Your Third Application: StringMethods

Our last application this month demonstrates some of the `String` methods. As you go through this program, take a moment to go through the documentation that came with your development environment. In particular, look for the file `java.lang.String.html`. As you'll see as you learn more about Java, all the Java classes are part of some larger collection of classes. These collections take the form of **packages**. For example, the `String` class (along with the rest of the "built-in" Java types) are part of the `java.lang` package. To use a package, you use a mechanism similar to the `#include`. This mechanism is the `import` statement. We'll learn about the `import` statement in next month's column. The one package you automatically have access to is `java.lang` and so you don't need to `import` it to get access to the `String` class.

The file `java.lang.String.html` contains a complete description of the variables and methods that make up the `String` class. Use your Web browser to open this html file and look over the class.

Here's the stringMethods source code...

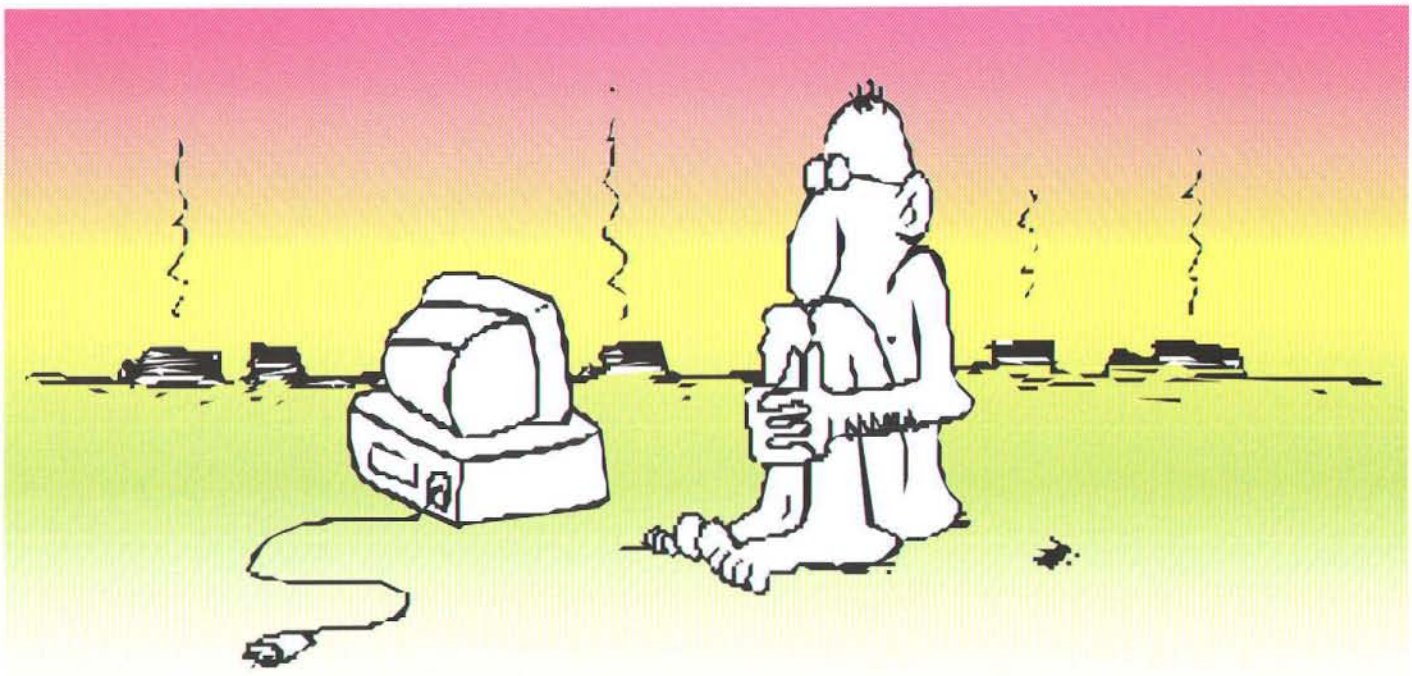
```
public class stringMethods
{
  public static void main(String argv[])
  {
    char myArray[] = {'a', 'b', 'c', 'd', 'e'};
    java.lang.String  string = new String( myArray );

    System.out.println( "string: " + string );
    System.out.println( "string[2]: "
        + string.charAt( 2 ) );

    string = string.concat( string );
    System.out.println( "Doubled string: "
        + string );

    System.out.println( "Index of first 'x': "
        + string.indexOf( 'x' ) );

    int index = string.indexOf( 'e' );
    System.out.println( "Index of first 'e': "
        + index );

    if ( index >= 0 )
      System.out.println( "Index of second 'e': "
        + string.indexOf( 'e', index+1 ) );

    System.out.println( "substring[2] to the end: "
        + string.substring( 2 ) );
    System.out.println( "substring[2] up to string[4]: "
        + string.substring( 2, 4 ) );

    string = string.replace( 'c', 'x' );
    System.out.println( "Replace 'c' with 'x': "
        + string );

    System.out.println( "Display as upper case: "
        + string.toUpperCase() );
  }
}
```

The first two lines show you yet another way to create a new `String`. As you'll see when you read through `java.lang.String.html`, there are several versions of the `String` constructor. Just like C++, Java supports function overloading, allowing you to create multiple versions of the same function, as long as each version has a unique signature (the function name combined with the parameter list).

This line defines an array of `chars` and initializes the array with the characters a through e:

```
char myArray[] = {'a', 'b', 'c', 'd', 'e'};
```

This line uses `new` to define the new `String` object, using the `char` array to initialize the `String` to the string "abcde". Notice that we refer to `java.lang.String` instead of just `String`. The two terms are equivalent. Since the `java.lang` package is automatically included, the `java.lang` prefix is not needed.



Do I have to use a PC to port Windows applications to a Mac?

WILLOWS TWIN APIW for Macintosh

Windows API tool box extension for Macintosh developers

- Multi-platform implementation of the Windows API
- Works with Macintosh tools and compilers
- Native library optimized for PowerPC or 68K
- Work within the IDE of your choice

Call now: (408) 777-1820

Visit our web site: http://www.willows.com

Come and see us at MacWorld!

WILLOWS SOFTWARE

Willows Software, Inc. 12950 Saratoga Avenue, Suite A, Saratoga, CA 95070

```
java.lang.String  string = new String( myArray );
```

The next line prints this string. `System` is actually `java.lang.System`. `java.lang.System` features a variable called `out` which features methods called `print()` and `println()`. We could have referred to `java.lang.System.out.println()` but, again, the `java.lang` is assumed.

```
System.out.println( "string: " + string );
```

Here's the output:

```
string: abcde
```

This next `println()` calls the `String` method `charAt()`. `charAt()` returns the nth character in the `String`.

```
System.out.println( "string[2]: "
    + string.charAt( 2 ) );
```

Here's the output. Note that Java strings are 0-based, just like C and C++.

```
string[2]: c
```

The `concat()` method appends its parameter to the end of the current object. In this case, we `concat()` `string` on the end of `string`, storing the result in `string`, then print the newly doubled `string`.

```
string = string.concat( string );
System.out.println( "Doubled string: "
    + string );
```

Here's the output...

```
Doubled string: abcdeabcde
```

The method `indexOf()` searches the string for the specified character, returning either an index into the string or the value -1.

```
System.out.println( "Index of first 'x': "
    + string.indexOf( 'x' ) );
```

Here's the output. Since the string doesn't contain an 'x', `indexOf()` returned -1.

```
Index of first 'x': -1
```

The next lines of code searches for the first 'e' in the String, storing the index in the variable `index`, then printing the index.

```
int index = string.indexOf( 'e' );
System.out.println( "Index of first 'e': "
    + index );
```

Here's the output:

```
Index of first 'e': 4
```

Next, assuming the index was not negative, we use a second version of `indexOf()` which takes a second parameter. This second parameter tells `indexOf()` where to start its search in the string, allowing us to search the string for a second 'e'.

```
if ( index >= 0 )
  System.out.println( "Index of second 'e': "
    + string.indexOf( 'e', index+1 ) );
```

Here's the output:

```
Index of second 'e': 9
```

Next, the `substring()` method is called. `substring()` takes an index into the string and returns a string that runs from the index to the end of the string.

```
System.out.println( "substring[2] to the end: "
    + string.substring( 2 ) );
```

Here's the output:

```
substring[2] to the end: cdeabcde
```

Next, we call a second version of `substring()`. This one takes a second parameter, an index that marks the end of the substring.

```
System.out.println( "substring[2] up to string[4]: "
    + string.substring( 2, 4 ) );
```

Here's the output:

```
substring[2] up to string[4]: cd
```

`replace()` replaces all occurrences of char 1 with char 2 in the string, then prints the result.

```
string = string.replace( 'c', 'x' );
System.out.println( "Replace 'c' with 'x': "
    + string );
```

Here's the result:

```
Replace 'c' with 'x': abxdeabxde
```

`toUpperCase()` replaces all lower case letters in the string with their upper case equivalents.

```
System.out.println( "Display as upper case: "
    + string.toUpperCase() );
```

Here's the result:

```
Display as upper case: ABXDEABXDE
```

Till Next Month...

Next month, we'll create our first true applet and learn a bit about the advanced windowing toolkit (AWT). Till then, take some time to read through some of the `.html` files that came with your development environment.

The Ultimate Database Engine To Make Your Applications Fly...

Lightweight, Powerful, and Fast!

Omniscience ORDBMS (Object Relational Database Management System) offers a small footprint, high performance, and conformance to standards for your database applications.

A small footprint allows you to deploy corporate data and applications in the field for mobile computing and PDAs (personal digital assistants), or in groupware environments. Omniscience is known by customers as having the smallest footprint (1/2 MB).

Omniscience ORDBMS is unbeatable on Windows when it comes to performance. Compared to other PC RDBMS and ODBMS products, Omniscience is by far the fastest database. Seeing is believing — take a look at our web site for detailed information regarding our benchmark: http://www.omniscience.com.

And why standards? Standards bring security to your investment in application development. Users value Omniscience's dedication to standards such as SQL-92, ODBC 2.0, and ODMG-93 1.1.

Conformance to standards offers you the freedom to select development tools like Oracle Power Objects, PowerBuilder, Delphi, Microsoft Visual Basic, Microsoft Visual C++, etc. You choose!

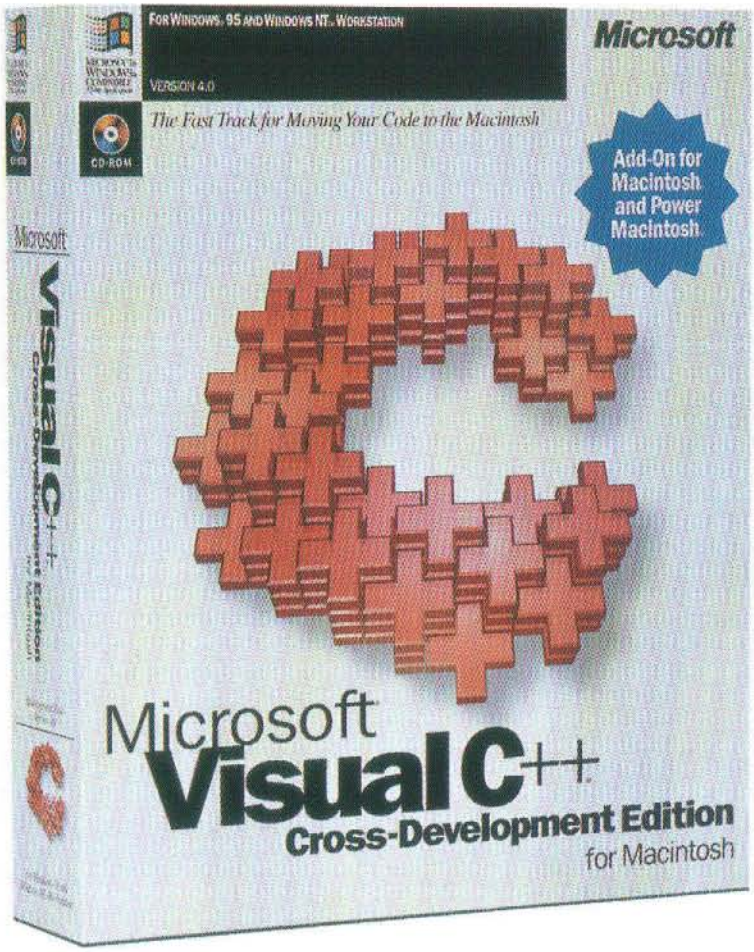
ORACLE Business Alliance Programme

Omniscience is now available for Windows 3.1x, Windows 95, Windows NT, Macintosh, and Unix. Omniscience supports both C and C++ language bindings.

Omniscience ORDBMS has many more robust features to offer but you can sign on to our web site at http://www.omniscience.com for more details. Also email or call us today for a copy of our technical white paper at info@oot.com or 408-562-0750.

Omniscience®

Omniscience Object Technology, Inc. 3080 Olcott Street • Suite 100C • Santa Clara, CA 95054
408-562-0750 • Fax 408-562-0757 • Email info@oot.com • http://www.omniscience.com



Bring out the best in both worlds.

Using just one toolset to target multiple platforms, you can increase your user base and knock months off your development time. With Microsoft® Visual C++™ Cross-Development Edition for the Macintosh*, you can reuse up to 90% of your code from your Windows-based applications to create the same applications for the Macintosh®, with the familiar Macintosh interface.

Features Include:

- *Port Windows®-based features* easily to the Macintosh, including multiple document interface, print preview, and floating toolbars. The Windows Portability Library does the work for you.
- *True native Macintosh programming*: Program directly to the System 7 API to take advantage of unique Macintosh features such as Publish and Subscribe.
- *Recompile for the Mac™*: Target both the Windows and Macintosh operating systems using just one set of source code written to Microsoft Foundation Classes and the Microsoft Win32® API.
- *Power Macintosh™ Support*: Including a fully optimizing compiler and integrated debugger that adapts to the PowerPC™ chip when displaying information in the register contents and disassembly windows.
- *Code Fragment Support*: for the Power Macintosh which translates to better code sharing among applications and faster, trimmer applications with very little development overhead.
- *OLE support*: You can now easily port your Win32 and MFC OLE applications to the Macintosh using OLE for the Macintosh version 2.06.
- *ODBC support*: You can call ODBC directly or use it through the appropriate MFC classes.

For more information: http://www.microsoft.com/visualc or call (800)719-5577.

Microsoft

WHERE DO YOU WANT TO GO TODAY?™

*This add-on toolset requires Visual C++ for Intel® platforms, available separately.

By Mark Alldritt

Improving AppleScript Performance

Avoiding the AppleSloth blues

Why Is AppleScript So Slow, Anyway?

This is the question our customers ask most frequently, and one to which there is no simple answer. The reason given by Apple is that the Apple event mechanism used by AppleScript to communicate with other applications has serious performance limitations under System 7. In my tests, there clearly are limits to the number of round-trip Apple events you can push through the system per second. However, there's something else wrong, apparently, because even AppleScript scripts which don't generate any Apple events do not seem to perform as well as similar scripts written in Frontier.

This article provides a number of tips for improving the performance of your scripts. Depending on the kind of scripts you are working on, these improvements in performance may turn out to be dramatic.

Improving Compile Time Performance

Regardless of what script editor you are using, you are bound to get to the point where you wish scripts would compile faster. Here are some tips for keeping compile times as short as possible:

- **Buy Speed Doubler if you are using a PowerPC-based Macintosh**. My customers have reported a 2 to 4 times improvement in AppleScript performance with Speed Doubler installed.

- **Keep the number of scripting additions to a minimum**. Having too many scripting additions slows AppleScript's load time, since it must process that many more files in the Scripting Additions folder. Also, more scripting additions means more event and class names must be parsed from `'aete'` resources and added to the AppleScript symbol table.

  A side benefit of keeping the number of scripting additions to a minimum is that you are less likely to run into terminology conflicts. This happens when a term is defined by two different scripting additions.

- **Split up your scripting projects**. Use the `load script` command to load pre-compiled portions of your script project. If you're using Script Editor, Script Wizard, FaceSpan or Scripter, it's probably best to store your pre-compiled scripts in a known location so that you can use simple Load Script statements:

```
property lib1 : load script alias "My HD:Script
Libraries:Lib1"
```

  Alternatively, you can store your libraries in the same folder as your script editor, and load them with the following command:

```
property lib1 : load script "Lib1"
```

  If you're using Script Debugger, you can use its `path to me` handling to store your libraries in a folder with your script:

```
property projectPath :¬
   «property ctnr» of item (path to me) of ¬
   application "Finder" as string
property lib1 : load script (projectPath & "Lib1")
```

Mark Alldritt is the president of Late Night Software Ltd. He's the developer of Script Debugger, a development environment for AppleScript. He's also the author of a number of other AppleScript-related products and tools such as Scheduler, Script Tools, File Visibility and MacPPP Control. You can reach him at alldritt@wimsey.com.

Once you've loaded your library, you can access properties and call handlers stored within it like this:

```
propertyName of lib1

handlerName(p1, p2, p3) of lib1
```

- **Don't overdo AppleScript formatting**. The formatted text feature of AppleScript is the only really good way to understand how AppleScript has interpreted your script during compilation. Unfortunately, the more formats you use, the slower your compiles will be. The slowdown is caused when your script editor is forced to cope with growing numbers of style runs as it displays your script following a compile. The answer is to keep the number of formats you use to a minimum.

  This point becomes more important as the size of your scripts grow. For short scripts, it's not really an issue, but for scripts larger than about 100K it can represent as much as a couple of minutes of the total compile time. A PowerPC-native script editor helps here (Script Debugger, Script Wizard), but it's always going to be an issue.

- **Keep background activity to a minimum**. All the script editors allow some level of background activity during compiles. If you keep this to a minimum, your compiles will be faster. Avoid having the Finder perform folder size calculations, turn off personal file sharing, etc. If you're using Scripter, turn off its "allow background compiles" preferences option. This speeds compiles dramatically.

Improving Run Time Performance

Probably the most frustrating part of using AppleScript is watching its seemingly slow progress through your script. Here are some tips and ideas for making scripts run faster.

- **Install Speed Doubler if you are on a PowerPC**. See above.

- **Keep the number of Apple events your script generates to a minimum**. Under System 7, Apple event performance is poor. A System 7 application can process at most 60 events per second, with typical performance more like 20–40 events per second. Reducing the number of Apple events will significantly speed up your script.

  The most common mistake is to iterate over application objects using AppleScript. Instead, use "whose" clauses to get the server application to do the iteration for you. (Note: not all scriptable applications support this – complain to the

WHY NOT USE THE BEST?

Companies like Adobe, Claris, Symantec, Metrowerks, Netscape and hundreds of others recognize the unsurpassed quality of Developer VISE and have switched to MindVision for their installation needs. In fact, every day MindVision sells more installers than all other competitors combined. Don't take our word for it. Ask any of our hundreds of satisfied customers why they switched from the competition to MindVision's Developer VISE.

We didn't stop there.
Announcing...

INSTALLER VISE 4.0

The New Industry Standard for Software Installers.

Find, move, rename, or delete any file. Create hierarchical packages. Build one installer for multiple languages. Build installers with unparalleled ease. Attach AppleScripts to your installation. New archive features allow you to build CD-ROM installers quickly and easily. It may not slice and dice, but in addition to installing it also removes. And much, much more.

And Introducing...

UPDATER VISE 1.1

The Most Advanced Updater Available.

Updater VISE is the only product that can update multiple versions of your 68K, PowerPC, and Fat application from ONE updater. Updaters are extremely compact and totally secure, allowing for easy online distribution. And, combined with Installer VISE you can update any number of different files giving you a total software delivery solution.

Maybe it's time to stop and take a look at MindVision's software delivery solutions.
Visit our web site for full information,
including fully-functional demos.

MindVision Software

www.mindvision.com

Email: info@mindvision.com • Voice: (402) 477-3269 • Fax: (402) 477-1395

VISE SOFTWARE DELIVERY SYSTEM

developers and let them know you want full support for the "object model".)

The slow way:

```
tell application "Finder"
    repeat with aItem in (items of first window)
        if name of aItem ends with ".temp" then
            delete aItem
        end
    end
end
```

The fast way:

```
tell application "Finder"
    delete (items whose name ends with ".temp") ¬
        of first window
end
```

The next error people commonly make is to repeatedly ask for the value of properties. For example, you might do the following:

```
tell application "Finder"
    repeat with aItem in items of window 1
        if name of aItem ends with ".c" or ¬
            name of aItem ends with ".h" then
            -- do something with .c and .h files
        end
    end
end
```

This script fragment gets the value of the application's name property twice. A faster method is to get the name property once:

```
tell application "Finder"
    repeat with aItem in items of window 1
        set theFileName to name of aItem
        if theFileName ends with ".c" or ¬
            theFileName ends with ".h" then
            -- do something with .c and .h files
        end
    end
end
```

- **Avoid using AppleScript's object-oriented features.** AppleScript supports object-oriented programming to a great extent. I've found that handler calls within object instances tend to be slow (the first handler call is the worst). Use these features only if they are justified.

- **Use properties to store pre-compiled values.** If there are values which are calculated once at the beginning of your script, consider putting these calculations into property definitions. By doing this, you cause the calculations to be performed at compile time rather than run time.

  This technique can be particularly dramatic if you are using scripting additions to help with the calculations, since the Apple events used to invoke the scripting additions are generated at compile time.

- **Use scripting additions.** Use scripting additions for complex list and text manipulation. The ACME Script Widgets and other collections of scripting additions offer tools for text parsing, tokenization, list manipulation and other functions. It's almost always going to be faster to use a scripting addition rather than an AppleScript loop to work with these data structures.

- **Execute scripting addition commands within your applet's process.** As stated above, Apple events sent to other processes for handling are slow (20–60 events/sec); but events processed within the script applet are handled much faster, since they are dispatched without involving interprocess communications and process context switches.

  So, in the following example, the `current date` command (which is a scripting addition command) is executed within the Scriptable Text Editor's process rather than the applet:

```
tell application "Scriptable Text Editor"
        make new line at end of first window ¬
                with data (current date as string)
end
```

  An improvement in speed would result if you did it this way:

```
set theData to current date as string
tell application "Scriptable Text Editor"
        make new line at end of first window ¬
                with data theData
end
```

  If you are deep within a tell block, you could deal with the problem this way:

```
tell application "Scriptable Text Editor"
        ...
        make new paragraph at end of first window with ¬
                data ((current date) of me as string)
        ...
end
```

- **Use the right tool for the job.** Sometimes you can get dramatic speed improvements by switching tools. For example, if you're doing text manipulation, consider using MacPerl, or BBEdit, or a special-purpose scripting addition. If you have several tools available, try using each of them to perform time-consuming parts of your script. See which one is best suited and performs best.

- **Use the right object accessing method.** The Object Model is a powerful thing, and offers great flexibility for the script writer. The problem is that it allows you to easily create object specifications that can be very difficult for the server application to resolve.

  You can sometimes improve performance greatly by changing the way you access objects. For example, text editors generally maintain internal data structures to help

them identify the beginning of lines. As a result, line-based object specifications are resolved much faster than word- or paragraph-based specifiers, because the application can use the data it already has to speed things up.

Try using different ways of forming your object specifiers and see which ones are fastest.

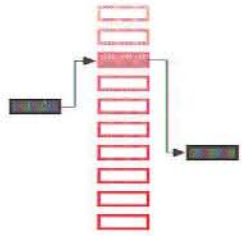
- **Code time-consuming logic in OSAXen**. As a last resort, you can code time-consuming portions of your script as a scripting addition. *MacTech Magazine* 10.1 featured an article on just this subject, and you can look at it online at http://web.xplain.com/mactech.com/Articles/Vol.10/10.01/Extend-Applescript.bhtml. Documentation on writing scripting additions can also be found in Apple's *AppleScript Scripting Additions Guide.*

The Final Frontier

The applets (standalone AppleScript applications) produced by Script Editor, Script Debugger, Script Wizard and Scripter all rely on the standard applet glue code provided by Apple. This glue code provides a framework in which your script executes. The standard glue code attempts to be a really good MacOS citizen. To this end, it yields the CPU to other processes very frequently. One way to improve the performance of your scripts is to write your own glue code, and yield the CPU less frequently, or possibly not at all. This change, depending on the frequency of yields, can cause speed improvements of up to 25%.

The Future

The future of AppleScript is quite bright. Following a period of almost non-existent support from Apple, there is to be a new version (1.2) released sometime this year. This new version will be PowerPC-native, and is expected to offer performance comparable to that of Frontier. Beyond the 1.2 release, there's Copland. There is reason to expect that under Copland, the AppleEvent Manager will offer dramatic performance improvements; supposedly, Apple events will be dispatched directly to your handlers, not to your app's event loop, thereby removing a serious speed bottleneck. And, there are rumours of a completely re-architected post-Copland AppleScript 2.0. In the meantime, AppleScript continues to prove its worth, and the techniques in this article should help you see some script speed improvements now.



By Peter N Lewis, Perth, Australia

Memory Madness

Getting a grip on handles

Introduction

Using the Memory Manager's handles effectively is a very important part of programming on the Macintosh. Using them badly is also a prime cause of many system crashes. This article describes the difference between memory and resource handles, and operations on them that you should use or avoid. After a brief introduction to the Macintosh memory model, I'll list a bunch of rules or guidelines along with the reasoning behind them [1].

Background

If you know the basics of pointers, handles, resources and so forth, you can skip on to the next section. I'll try to keep this reasonably simple, but if you don't know what a pointer is, you should probably skip this whole article and go buy a beginner's guide to programming.

On the Macintosh, the Memory Manager (the system software which keeps track of all memory) divides the available RAM (including virtual memory) into chunks called "heap zones". When your application is launched, you are allocated some memory as a single heap zone (the size of this heap is defined by the "Get Info" size you set in the Finder or originally in the `SIZE` resource of your application). You use Memory Manager routines to allocate or release memory in your application zone, either directly with routines like `NewPtr` and `NewHandle`, or indirectly with routines like `GetResource`. Lots of routines allocate memory in your heap, but this article is going to concentrate only on the Memory Manager and the Resource Manager. Similarly, you can allocate memory and create your own heap zones, and you can allocate memory outside your own heap zone (either in the System Zone or as Temporary Memory), but I'll stick to just your application zone.

The simplest way to allocate memory is to use the `NewPtr` routine. You tell it how many bytes you want, and it returns a pointer to the memory, or `nil` (`NULL`) if there was not enough memory available. One problem with pointers is that you often can't resize them. Even if there is lots of free space left in the heap, there might be something allocated in the memory just after the pointer, so resizing it quickly becomes impossible. A solution to this problem is to use handles.

A handle is a Memory Manager structure which is basically a pointer to a pointer (the first pointer is a handle, the second one is called a master pointer). Your code remembers the handle, and then, to resize it, the Memory Manager can free the master pointer and allocate new space anywhere in the heap. You allocate a handle using `NewHandle` and you release it using `DisposeHandle`.

One of the most common times that you will use handles is when dealing with resources. All resources on the Macintosh are allocated as handles. You allocate a resource handle using `GetResource` and you release it using `ReleaseResource` or by closing the resource file.

Real Pointers and Handles

The Mac operating system is not known for robust APIs (particularly those APIs that have been with us since 1984). If you pass invalid parameters to the operating system, at best it will do nothing useful, at worst it will crash the system and

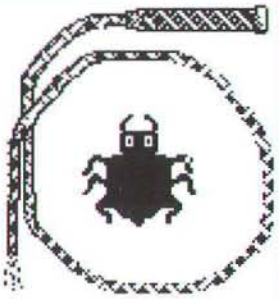
Peter N Lewis is a successful shareware author. He founded Stairways Software Pty Ltd in 1995 and specializes in TCP/IP products but has been known to diversify into other areas.

Gives you the Information to *Program your Best!*

The Debugger V2 & MacNosy

by Steve Jasik

Information



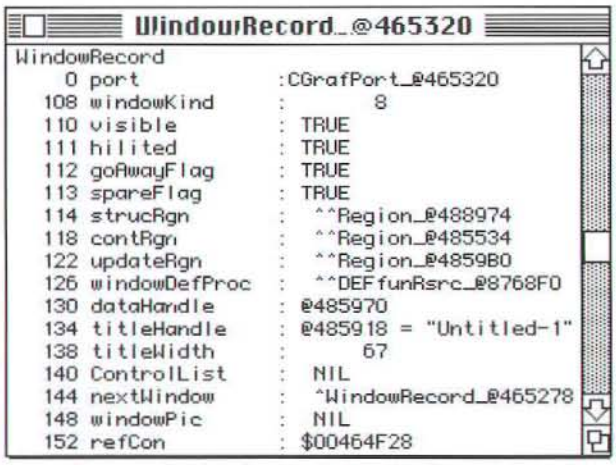
Control

For over 10 years Steve Jasik has brought Macintosh Developers the superior debugger. In addition to the software, Steve provides timely support to his customers. Inspect the feature list and see why so many developers use it. In addition to basic features such as stepping and breakpoints, the Jasik debugger has many advanced features such as heap display, a fast memory watch, Soft-MMU, jump tracing, ... to help you track down the elusive bugs that other debuggers won't help you find.

In additon, the package includes **MacNosy,** a global interactive disassembler that enables one to recover the source code of any Mac application or from the ROM.

When you compare features of the different debuggers, note that *only one* has all the below features to help you get your job done, and *only one* has MacNosy to help you debug any program in a full system environment symbolically!

"The Debugger" is the debugger of choice at: Adobe, Apple, Claris, Kodak, Metrowerks, etc.



An example of a structured data display window with hypertext links to substructures

Its Features Include:

- **Source level debugging for Metrowerks & MPW** compiled programs (C++, C, Pascal, Fortran, ...).
- **Symbolic Debugging** of *any* Macintosh program, ROM, Plug-In or *code resource* (DRVRs, XCMDs, INITs, PDEFs..), ASLM & **CFM LIbraries, OpenDoc parts**, GX Drivers, ...
- **Simultaneous Symbolic debugging of multiple "tasks"**
- **Object Inspector for MacApp 3 & PowerPlant programs**
- **Source level debugging of Symantec C™ projects**
- **Includes a program (CoverTest) to interactively do Code Coverage analysis for SQA testing, etc.**
- **Fast Software Watchpoint** to find clobbered variables.
- **Soft MMU** ("heap centric" bounds checking) for PowerPC programs to find out of bounds references.
- **Jump Trace** to find where a program takes a "wild jump"
- **Sophisticated error check algorithms** such as Trap Discipline (Argument Checking), Heap Scramble and Heap Check to detect errors before they become disasters
- **Structured display** of data (hypertext) with user definable structures while debugging
- **Conditional breakpoints** to filter redundant information
- **Continuous Animated Step Mode** to watch your program execute instruction by instruction
- **Detailed symbolic disassembly for both 680x0 and PowerPC** with symbol names, labels, cross ref maps, – make it possible to ferret out the secrets of the ROM, etc.
- **"Training Wheels"** for the PowerPC disassembler to help you learn the opcodes

The Debugger V2 & MacNosy: $350

Runs on all Macs. Call For Group prices or Updates. Visa/MC Accepted.

Available from: Jasik, APDA or ComputerWare (800-326-0092).

Jasik Designs • 343 Trenton Way, Menlo Park, California 94025 • (415) 322-1386

URL: http://www.jasik.com e-mail: macnosy@jasik.com

corrupt people's data. So it is very important to ensure that you are always playing nice with the Memory Manager.

1 Always check whether a memory allocation returns nil.

I know you've heard this a thousand times before, but it can't be repeated enough. `NewPtr`, `NewHandle`, `GetResource`, and so forth can all fail and return `nil`. If your code blissfully ignores this fact, you are going to go down in flames. It is especially important not to pass the `nil` returned by `NewPtr` to `DisposePtr`; that is a sure way to die (passing `nil` to `DisposeHandle` on the other hand is perfectly safe). See the section on writing `GrowZone` procedures for a way of reducing the stringency of this rule a little bit.

2 Never use fake pointers or handles.

A Memory Manager pointer is more than just any old pointer you get using `&variable` (or `@variable` in Pascal). Similarly a handle is more than just any old pointer to any other pointer. These are called "fake" pointers or handles; real pointers and handles must be allocated by the Memory Manager. If you try something really silly like `DisposePtr(&variable)`, very bad things will happen. Also, the master pointer is not a real pointer, so never do anything like:

```
h := NewHandle( 10 );
SetPtrSize( h^, 20 );
```

3 Never mess with the master pointer.

As described above, a handle is a pointer into a block of Memory Manager memory which contains a master pointer pointing to your data. You must never modify the master pointer in any way. For example, never ever do something like this:

```
h = NewHandle( 10 );
for ( int i = 0; i < 10; i++ ) {
  *(*h)++ = 0;
}
```

Not only will this corrupt the heap, but it would be much simpler to just use `NewHandleClear`.

4 Always colour between the lines.

It probably goes without saying (but I'll say it anyway): don't write to any memory outside the handle or pointer's allocated space. If you allocate 10 bytes (and it succeeds) make sure you only write to the first 10 bytes pointed to by the pointer or master pointer.

5 Always dispose of memory exactly once.

Once you dispose of memory (for example, by calling `DisposePtr`, `DisposeHandle`, or `ReleaseResource`, or by closing the resource file), the pointer or handle you had is no longer valid. You must not use it for any purpose, and especially you must not dispose of it again! Doing so will corrupt the heap.

Locking and Purging

The important feature about handles is the ability for the data to move around in memory so that it can be resized. Unfortunately, this also introduces a lot of possible problems, since the Memory Manager can move the memory any time it is called, directly or indirectly, by you or anyone else (your handle will stay valid, but the master pointer will change).

6 Always lock your handles when you dereference.

You must lock a handle (using `HLock`) any time you dereference it (that is, any time you remember the master pointer in another variable or pass it to another procedure, or use a `with h^ do` statement in Pascal), unless you are absolutely sure you are not going to call any routines that may move memory.

There is a list of routines that may move memory, which would seem to imply that there is a list of routines that must not move memory. But since not everyone has read both lists, and since many of those who have not read them have spent their time more productively by writing System Extensions that patch routines that are not suppose to move memory so that they now do move memory, about the best course of action is to assume that *every* system routine that you call may move memory. The only exceptions I would make to this are routines called at interrupt level (since you are not allowed to call any Memory Manager routines at interrupt level), and `BlockMove` and `BlockMoveData`.

It is safer to lock and then unlock a handle than it is to find out the hard way that a routine sometimes moves memory – these bugs are basically impossible to track down. But on the other hand you don't have to go completely insane either – if you don't call any routines at all, then the memory cannot move. So it is perfectly safe to scan a handle looking for a linefeed, for example.

7 Use HGetState / HLock / HSetState instead of HLock / HUnlock.

Imagine you write a procedure that needs to lock a handle; for example:

```
procedure DontDemonstrateSetupData( data: Handle );
begin
  HLock( data );
  DoStuff( data^, GetHandleSize( data ) );
  HUnlock( data );
end;
```

If you now call this routine after locking a handle, it will cheerfully unlock it for you, with potentially terrifying results. Instead of that, you should use `HGetState` to preserve and restore the state.

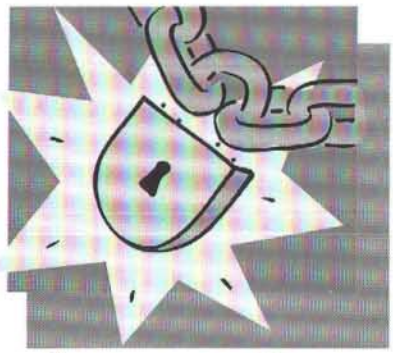
```
procedure DemonstrateInitializeData( data: Handle );
  var
    state: SignedByte;
begin
  state := HGetState( data );
  HLock( data );
```

dtF The Relational Database System

dtF v1.6 including numerous SQL enhancements and ODBC driver available!

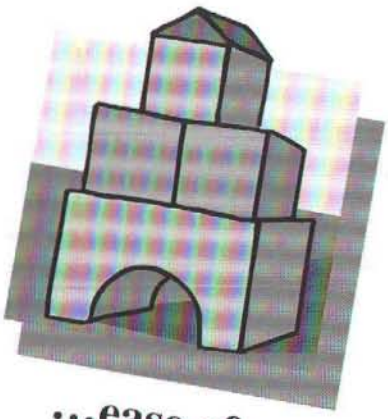
When it comes to performance **dtF** is in a class all its own. **dtF** utilizes a proprietary query optimization and caching scheme to obtain unparalleled performance. If you are in need of a quantum leap in performance, use the **dtF** native PowerPC stand-alone or client/server version.

...performance



Full transaction control and automatic level-3 error recovery guarantee maximum data protection even after sudden system crashes. **dtF** databases are compressed and encrypted to protect against all unauthorized access, even disk editors. Implement sophisticated table and operation level security with **dtF**'s password features.

...data security



...ease of use

With **dtF**'s unique direct integration technology, the **dtF** database system is fully contained in your application, enabling you to build double-clickable database applications. Never worry about missing or conflicting INITs or drivers. **dtF**'s native API is compact, easy-to-use and integrates seamlessly with all major development environments on the Macintosh. **dtF**'s high performance SQL, integrated data dictionary, security features, automatic index selection, query optimization and error recovery allow you to concentrate on creating great database applications instead of messing around with the internals of the database system.

Stand-alone applications built with dtF are royalty-free!

dtF is available for Macintosh System 7.x (68K and native PowerPC). **dtF** supports MPW C/C++, Symantec C/C++, Metrowerks CodeWarrior (all compilers 68K and native PPC). **dtF** is fully OpenDoc™ compatible. Separate versions for use with HyperCard 2.x, SuperCard 2.x, Smalltalk-Agents and Pictorius Peregrine are provided. AppleScript interface via DataScript™ for **dtF** from General Knowledge. **dtF** supports cross-platform development on Windows 3.11, Windows 95 and OS/2.

dtF Americas, Inc.
19672 Stevens Creek Blvd.,
Suite 128
Cupertino, CA 95014
USA

Phone: (800) DTF-1790
Fax: (510) 828-8755
AppleLink: DTF.AMERICA
Internet: dtF.america@applelink.apple.com

dtF The Relational Database System

```
  DoStuff( data^, GetHandleSize( data ) );
  HSetState( data, state );
end;
```

An alternative approach is to assume that all handles are unlocked, and any routine can unlock a handle. So after any call to a procedure you have to relock and re-dereference the handle.

8 Watch out for purgeable resources.

If you tell the Memory Manager that a handle is purgeable (either using `HPurge` or setting a resource's purge bit using ResEdit), then the memory may be released any time it could be moved (if you lock the handle it will not be purged, so there is generally no need to call both `HLock` and `HNoPurge`). The normal case for using purgeable handles is when you make a resource handle purgeable; then you can load the resource using `GetResource` (or `GetIndString` or whatever) and not bother releasing it. The Resource Manager will release it automatically if the resource file is closed, and the Memory Manager will release it if you run low on memory.

The best way to deal with purgeable resources is to always call `GetResource` when you need the resource, and then use either `HGetState/HLock/HSetState` (as described in Item 7) or `HGetState/HNoPurge/HSetState` to ensure that the data is not released until after you are finished with it. I would recommend that you never use `HNoPurge` and `HPurge`; instead, a resource handle should always remain either purgeable or non-purgeable. In the former case you should be careful to always reload the resource (using `GetResource`, or, if you remembered the handle, using `LoadResource`) and to lock it while it is in use.

Memory vs. Resource Handles

There are subtle but important differences between a memory handle (one you get by calling `NewHandle`) and a resource handle (that you get by calling `GetResource`).

9 Match NewHandle / DisposeHandle and GetResource / ReleaseResource.

You must make sure you always release memory handles using `DisposeHandle` and release resource handles using `ReleaseResource`. This is because the resource manager keeps extra information related to resource handles, so you must use `ReleaseResource` to ensure that the resource manager knows that the resource handle is no longer valid.

10 Always know whether a handle is a resource handle or a memory handle.

One consequence of Item 9 is that you must always know whether a handle is a resource handle or a memory handle in order to dispose of it. For instance, you should never have code that looks like this:

```
h = GetResource( 'STR ', 128 );
if ( h == NULL ) {
  h = NewString( "\pHello" );
}
```

At the end of this sequence, you don't know whether `h` is a resource handle or a memory handle, so how can you dispose of it properly? The simple solution in this case is to ensure that at the end of the snippet we are left with a memory handle no matter where we got the memory from. You can do this by using `DetachResource` to change the resource handle returned by `GetResource` into a memory handle:

```
h = GetResource( 'STR ', 128 );
if ( h == NULL ) {
  h = NewString( "\pHello" );
} else {
  DetachResource( h );
}
```

We are now assured that we can later correctly use `DisposeHandle` to dispose of the memory.

It is possible to determine whether a handle is a resource handle or a memory handle, like this:

```
isresource := (HomeResFile( h ) <> -1);
```

but in general you should know what kind of handle you are dealing with. I suppose one solution would be to write a routine like this:

```
procedure DisposeAnything( var h: Handle );
begin
  if h <> nil then begin
    if HomeResFile( h ) <> -1 then begin
      ReleaseResource( h );
    end else begin
      DisposeHandle( h );
    end;
    h := nil;
  end;
end;
```

However, this is not a particularly efficient solution, since `HomeResFile` probably takes a fair amount of time to confirm whether a handle comes from a resource file or not.

11 Convert between resource and memory handles where appropriate.

As seen in Item 10, it is possible to convert a resource handle into a memory handle using `DetachResource`. You can also go in the other direction by adding a handle to a resource file using `AddResource`. Releasing the memory is not the only time you have to ensure that you know what kind of handle you have; you also cannot add a resource handle to a resource fork, so you will have to use `DetachResource` before calling `AddResource`, like this:

```
h = Get1Resource( 'STR ', 128 );
err = ResError();
if ( h != NULL ) {
  DetachResource( h );
  AddResource( h, 'STR ', 129, "\pteststring" );
  err = ResError();
}
```

12 CloseResFile releases all resources.

When you close a resource file, all the resources are automatically released. This means that any resource handles you have that came from that resource file are now invalid, so you must not use them (including not calling `ReleaseResource` or `DisposeHandle` on them). If you want to keep a resource handle around after you close the file, you must turn it into a memory handle by calling `DetachResource`. So, for example:

```
result := nil;
resfile := FSpOpenResFile( spec, fsRdPerm );
if resfile <> -1 then begin
  str1 := GetResource( 'STR ', 128 );
  str2 := GetResource( 'STR ', 129 );
  if (str1 <> nil) & (str2 <> nil) then begin
    if length(str1^^) > length(str2^^) then begin
      result := str1;
    end else begin
      result := str2;
    end;
    DetachResource( result );
  end;
  CloseResFile( resfile );
end;
```

At the end of this code, data is either `nil` or, assuming both string resources exist, data is a memory handle containing the longer of the two strings. There are a bunch of things to notice in this code. First, it defends against failing to open the resource file or failing to get the string resources. Next, it lets the Resource Manager release the resource handles (so for example, if `str1` is `nil`, `str2` will still be released by the Resource Manager when the file is closed). Also, the code is careful to detach the resource handle we wish to keep past the `CloseResFile` so that it is not automatically released.

Cool Memory Manager Routines

The Memory Manager provides a lot of neat routines for working with handles. Most of them you can duplicate yourself, but why waste time and introduce potential bugs when you can just get the OS to do it for you?

13 Use PtrAndHand and friends.

`PtrAndHand` appends a chunk of memory to the end of a handle. Similarly, `HandAndHand` appends a handle to another handle. `PtrToXHand` replaces a handle's data with new data. In all cases, the source data is unaffected and the destination handle (which must already be a valid handle) is resized appropriately and the new data is copied in. If the destination handle was a resource handle, it remains a resource handle (although it will only be written back if you call `ChangedResource`).

`PtrToHand` and `HandToHand` allocate a new handle and initialize its size and contents based on the input values. The resulting handle is always a memory handle even if the source handle was a resource handle.

Another nice thing about these calls is that they return OSErrs, so you don't have to call MemError.

PtrAndHand is really useful for building a handle from a sequence of input data (for example, you might have a handle to a text log, and you might append new lines to the log by using PtrAndHand).

14 Use Munger where appropriate.

Munger is my favourite Macintosh routine. It is a true power-geek tool. If you can master this routine you can amaze your friends with astonishing feats. Munger looks pretty complicated (it takes a handle, two pointers and three longs), and it does a bunch of almost unrelated things depending on the exact parameters you pass it. But once you get the hang of it, it is fairly easy to use.

```
function Munger(
    h: Handle; offset: longint; ptr1: Ptr; len1: longint;
    ptr2: Ptr; len2: longint) : longint;

pascal long Munger(
    Handle h, long offset, const void *ptr1, long len1,
    const void *ptr2, long len2);
```

Basically, what Munger (which rhymes with "plunger", according to *Inside Mac*) does, is to search and optionally modify a handle (the first parameter). The second parameter is an offset to start searching from (normally this is zero to start from the beginning of the handle). The next two parameters (ptr1 and len1) describe the data to search for (it is a byte search, so it is case-sensitive, and WorldScript-ignorant so you probably cannot use it for WorldScript text). One trick with the search parameters is that if you pass it nil for the pointer it will act as if it finds a match immediately (I'll give you an example below, so don't panic if you didn't quite follow that). The final two parameters (ptr2 and len2) describe the replacement data. The matched data will be replaced with this second chunk of memory, assuming that the pointer is not nil (one trick here is when you want to delete the found data, you need to pass a non-nil value with a zero size; I normally use the address of the source handle, but any non-nil value will do). The return value of Munger is either the offset of the matched data, or -1 if no match was found. Munger also sets MemError if it fails to resize the handle, so if you are inserting data you must check for an error. Okay, you are probably lost by now, so let's look at some examples.

In the first example, we will just use Munger to insert some text.

```
h = (Handle) NewString( "\pHello World!" );
(void) Munger( h, 7, nil, 0, (Ptr) "Cruel ", 6 );
err = MemError();
```

What this does is to match zero bytes at offset 7 (six characters plus the length byte) in the handle, and then to replace those zero bytes with six bytes of "Cruel ". (It is customary, where I come from, to make your first program in any new language print "Hello Cruel World!"; I'm not sure what that says about the people I hang out with, but I expect most programmers can see the logic in it). Don't forget to reset the pascal string length with:

```
(*h) = GetHandleSize( h ) - 1;
```

Alternatively, if you're having a good day, you might prefer to remove the "Cruel" like this:

```
h = (Handle) NewString( "\pHello Cruel World!" );
(void) Munger( h, 7, nil, 6, &h, 0 );
```

Starting from seven bytes into the handle, this matches any six bytes and replaces them with zero bytes (starting from &h, not that that matters much; all that matters is that &h isn't NULL). We don't need to check MemError after Munger because we are reducing the size of the handle, and the memory manager pretty much has to be able to cope with that (of course, we should have tested the handle returned by NewString to ensure that that succeeded!).

Alternatively, you might be having a really really good day, and want to replace "Cruel" with "Wonderful", like this:

```
h = (Handle) NewString( "\pHello Cruel World!" );
where = Munger(
     h, 0, (Ptr) "Cruel ", 6, (Ptr) "Wonderful ", 10 );
err = MemError();
```

This searches for the six bytes "Cruel " and replaces them with the ten bytes "Wonderful ". It returns the offset where "Cruel " was found (in this case, it will return 7).

If you just want to search for where the "Cruel" appears, you can do this:

```
h = (Handle) NewString( "\pHello Cruel World!" );
where = Munger( h, 0, (Ptr) "Cruel ", 6, nil, 0 );
if ( where >= 0 ) {
  printf( "Found at offset %ld\n", where );
}
```

The handle is not modified because `ptr2` is `NULL`.

GrowZones

The Macintosh system becomes very fragile when you run out of application memory. It is also very tedious to have to guard every single tiny memory allocation (including creating new objects and new empty handles, and so forth). One way to reduce the chance of bad things happening, and to let you relax a little bit, is to install a `GrowZone` routine. The Memory Manager calls this routine when it cannot meet a request for memory. It is very easy to write a simple `GrowZone` procedure by allocating some spare memory when your program starts up (say 20k); then, when the `GrowZone` routine is called, you simply deallocate this memory in the hope that the Memory Manager will now be able to meet the request. In the event loop you can check whether the memory has been released, and if so (and if you cannot reallocate your spare memory) then you can display an alert and quit gracefully. Also, when you make a large memory allocation, check that the allocation succeeded (i.e., the handle is not `nil`), but also check that the `GrowZone` memory is still available. If the `GrowZone` has fired and released its memory, you should dispose of the handle you just created and pretend that the memory allocation failed.

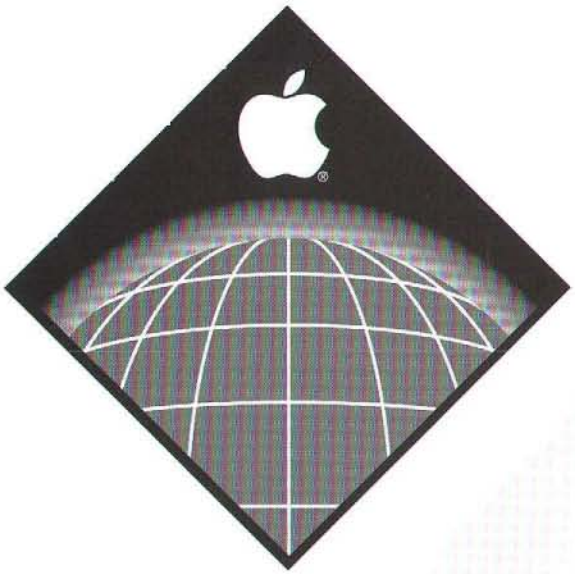
Conclusion

All of the above should be considered as guidelines – you should follow them unless you have really good reasons not to. Even if you follow all of them, you will sometimes run into trouble with some sort of heap corruption or memory problem. You are not completely on your own when this happens; there are tools that can help you. For example, the Debugging Modern Memory Manager can detect some of the problems caused by ignoring (or forgetting) the guidelines I've listed, and Even Better Bus Error can detect writes to `nil`. Before shipping any program, you should install the DMM and EBBE and stress-test your program – if you don't, you can be sure some of your users will, and you and they will both be much happier if you find these problems before you ship.

References

[1] Scott Meyers, *Effective C++* (Addison-Wesley: 1992). I use a very similar format to the one Scott uses in this excellent book.

[2] *Inside Macintosh: Memory.*



1996 Apple Worldwide Developers Conference

Here's a sample of the participating vendors at the 1996 Apple Worldwide Developers Conference.

Come see Apple's exciting third party development tools and products at the vendor exposition taking place during the WWDC May 14-16 in San Jose.

Adaptec

Adaptec, the company that made PCI and SCSI technology the standard in non-Mac systems, offers Apple Power Macintosh users the PowerDomain family of products and Remus RAID software.

Advantage

Advantage Memory Corporation is a first class manufacturer of computer memory upgrades and PC cards for PC's, notebooks, laser printers, and workstations. Our excellent reputation has been earned through reliable products and competitive pricing.

Apple Newton

Newton Toolkit is used to create Newton applications. Desktop Integration Libraries (DILs), are used to move data back and forth between a Mac OS or Windows application and a Newton PDA.

Ariston

Ariston Technologies is a developer and manufacturer of memory upgrades, SIMMs, DIMMs, PC DRAM cards, PCMCIA fax/modems, PCMCIA ethernet and combo cards, Video RAM and Cache cards for Mac's, PC's, File servers, Powerbooks, Laptops.

Be Inc.

The BeBox™ combines powerful hardware featuring multiple PowerPC™ processors, a portable object-oriented operating system, and advanced sound, graphic, video, and communication capabilities, to create a high performance system.

Brooktree

Brooktree Corporation will be showing the BtV® MediaStream™ solution providing new multimedia, gaming, videoconferencing, and on-line Web intercast opportunities. They will also show their software wavetable engine Wavestream.

CE

CE Software, Inc.'s QuickMail is the leading Mac OS messaging system. Key features: cross-platform clients, server & client based rules. Web Arranger is the only Internet Organizer for the Mac OS. Key features: URL validator, WWW page change notification.

Developer Depot™

A new mail order concept targeted towards Mac OS developers. Guaranteed low prices, world renowned service, and a great selection. See all the MacTech products.

Eastman Kodak

Eastman Kodak provides a range of award winning digital cameras, scanners, image conversion and color management services. Developer tools and program information will be available at the stand.

EveryWare Development Corp.

EveryWare Development Corp. is a leader in the Web authoring and site development marketplace. Tango provides Webmasters with a visual development environment. Web site redirects Web server log information to a SQL database.

Farallon

Farallon Computing Inc. is showcasing Netopia family of plug and play, high speed, ISDN Internet access products and services. Also displayed, FREE Internet collaboration applets offering real time messaging and screen-sharing over Internet.

Guideworks

Guideworks, LLC is a full-service Apple Guide developer. We develop guides, consult, and provide cross-platform help solutions.

Kaidan

Kaidan manufacturers of QuickTime VR photographic equipment, including QuickPan panoramic tripod heads (35mm, QuickTake, DC40, underwater, camcorders) and the Magellan family of object rigs (desktop & floorstanding).

Lotus Development Corp.

Lotus Notes enables you to communicate with colleagues, collaborate in teams, and coordinate strategic business processes, from within your company and extended throughout the World Wide Web.

MicroNet

MicroNet Technology, Inc. will be showing revolutionary new storage and transport systems for the Mac.

Mindvision

MindVision Software sells Installer VISE, the leading Macintosh installer. Also see Updater VISE for building patchers, TMON for debugging, and our new Installer VISE for Windows.

Powersoft

Powersoft Corporation has announced immediate availability of PowerBuilder on the Apple Macintosh platform. Designed for team development of high-performance client/server applications, PowerBuilder offers an object-oriented development environment.

Prosoft

Prosoft Labs offers complete SQA and hardware quality assurance tailored to the needs of today's Mac developer. We have over 10 years experience developing and testing and helped many companies deliver trouble-free products.

For a complete listing of participating vendors, please check out the WWDC web site at http://www.info.wwdc.carlson.com/

By Guy Kawasaki, Apple Fellow

Getting the Most out of WWDC

Every year, Apple hosts its World Wide Developer's Conference (WWDC) for five thousand or so raging, inexorable Macintosh evangelists. It's a not-to-be-missed event, but even when you subtract the value of the t-shirts you're going to get, you'll spend a pretty penny to attend. So here's some advice on how to get the most bang from your WWDC buck.

First, hunt down engineers and product managers. Forget trying to corner the bigwigs like Gilbert Amelio, Dave Nagel, or Heidi Roizen (or me) to tell them about your breakthrough product and to get their blessing. To reach the executive level, there are segment loaders that swap out sound bytes very efficiently. Instead, pursue contacts with engineers and product managers because they have real clout. They decide who gets – officially and unofficially – pre-release information, beta copies, and other goodies.

Second, when you do find an engineer or product manager, make the right pitch. If I had a nickel for every developer who told me that his or her product would save Apple, I could donate a color LCD projector to every Macintosh user group in the world and still have enough money left over to buy a NSX. What Apple employees want to hear is that you will sell tons of *your* Macintosh products. If you're doing good, we're doing good, so focus on doing what's right for your customer and your company. If you do this, Apple will be okay too.

Third, avoid people who are wearing ties. Someone who wears a tie to WWDC, no matter what his title, needs to rebuild his desktop file and zap his parameter RAM. WWDC is about communicating information, not projecting image. Anyone trying to project image has seriously misjudged the purpose of attending WWDC and is likely to waste your time. Better you be in your hotel room playing Marathon 2, than listening to buzzword-laced conversations.

Fourth, don't assume Apple's right hand knows what the left hand is doing, and don't take "no" for an answer. This is a general recommendation no matter when you're dealing with Apple. If you want something from Apple – a copy of Copland, a Macintosh ROM listing, or a waiver of a copyright lawsuit – just keep asking until you find someone who will give it to you. If one part of Apple says no, don't assume that the rest of Apple is aware of this decision and will fall in line. In fact, there are some parts of Apple who will intentionally say yes if you're turned down – certain Fellows, for example.

Fifth, make connections outside of the Apple employee circle. The most important contacts you can make may not be Apple employees but your colleagues at other software companies and industry organizations. You're cheating yourself if you're not in the hallways discussing:

- What compiler produces the fastest code?
- What's the best way to write cross-platform applications?
- Which distributors and retailers are open to Macintosh products?
- How many hits has your company's Web page generated?
- Did bundling with the Performas help long-term sales?
- Which software evangelists at Apple return phone calls?

Guy Kawasaki is an Apple Fellow and former Macintosh software evangelist. He runs a listserver for Macintosh believers. To join, send an email to evangelist@macway.com.

Sixth, skip any panel with more than four speakers. A good moderator, like a good tester, is hard to find. Most moderators ask too many people to join their panel, and then allocate equal air time to all. What good is a seven-person panel for a one hour session? As Andy Grove would say, do the math: 7 panelists × 5 minutes/panelist for introductory remarks = 35 minutes. Sessions always start 10 minutes late, so if you're lucky, you're left with a grand total of 15 minutes of discussion. The ideal panel has three people: two panelists with violently opposing positions, and one moderator.

Seventh, take in the cultural highlights of Silicon Valley. (Do you know the difference between yogurt and Silicon Valley? Yogurt has culture.) Three places you won't want to miss are ComputerWare, Fry's Electronics, and Computer Literacy. ComputerWare is a Macintosh-only store that stocks so much Macintosh hardware and software that your mouth will water. (One WWDC breakout should be a field trip to ComputerWare to illustrate what a Macintosh store can look like. I suggest you take pictures to show your local Egghead store manager.) If a programmer were given the task of creating heaven, Fry's Electronics would be the end product: Twinkies, chips (silicon and edible), computers, audio equipment, and software under one roof. Computer Literacy is the world's greatest computer and business book bookstore. You can kill two birds with one stone because both Computer Literacy and Fry's are located near each other only five miles from the San Jose Convention Center. If you go to this mecca, make the trip a grand slam by eating at Togo's Deli (I recommend the #16 sandwich).

Finally, bring the family and stay the weekend. If you stay on Saturday night, airfare is much cheaper, anyway. Within two hours of the Convention Center are San Francisco, Napa Valley, and Monterey (with the Power Macintosh 9500 of aquariums). I can see it now: "Michael Jordan, you've just won the NBA championship. What are you going to do next?" "I'm going to WWDC!"

Color Your Thoughts.

Introducing BBEdit 4.0

We are proud to announce the text editor you've been waiting for: BBEdit 4.0. This major new release features syntax-directed text coloring for today's popular programming languages.

(BBEdit recognizes C, C++, Pascal, Java, HTML, Rez, FORTRAN, assembly language, Perl, Tcl, ScriptX and GuideScript.)

An Impressive Palette.

In addition to the new syntax coloring, BBEdit offers a diverse array of tools for artists of all persuasions: navigation of source files via our integrated PopupFuncs™ technology, automatic linefeed translation, parenthesis balancing, auto-indent, and more. A powerful search engine is coupled to multi-file directory scanning and "Grep" pattern matching, giving you the ability to locate text in currently open windows, previously searched files, or project documents.

BBEdit's unique ability to compare folders and projects makes it a natural for figuring out what you did to your masterpiece during your latest fit of artistic inspiration.

Use Your Favorite Brush.

In addition to our core editing features, we've incorporated awareness of Projector, direct support for ToolServer, and seamless integration with THINK C/Symantec C++ (version 6.0 and later) or Metrowerks CodeWarrior (through the "ClickWarrior" extension).

The Ultimate in Expression.

To help you exercise your artistic license, BBEdit is scriptable with any OSA-compliant scripting system (including Frontier and AppleScript). You can also extend BBEdit with plug-in code modules (we provide information and examples to help).

Modern Art.

Since its first commercial release in 1993, BBEdit has defined the state of the art for capability, user interface, functionality, and customer support that others are still trying to emulate. BBEdit was the first editor to support ToolServer, THINK Reference, Toolbox Assistant, and THINK C / Symantec C++. We're still setting the pace for others to follow, as the first Mac text-editing solution to deliver on CD-ROM with on-board documentation, the first text editor to support the Internet Config system, and the first development tool with Apple Guide.

In addition to its distinguished portfolio, BBEdit supports the latest system software technologies: AppleScript, Macintosh Drag and Drop, PowerPC acceleration, and Quickdraw GX printing.

The Critics Rave.

We've painted a great picture, but don't take our word for it. Visit our Web site to see why critics and the general public have anointed BBEdit the grand master of text editors.

BBEdit. It Doesn't Suck.

Accelerated for Power Macintosh

Mac™OS

BMUG FALL 1994 CHOICE PRODUCT

BMUG SPRING 1995 CHOICE PRODUCT

BMUG Choice Product FALL 1995

Visit us at the Apple Worldwide Developer's Conference. San Jose, California. May 13-17, 1996.

SYMANTEC C++
for Power Macintosh

Find Out What The Critics Are Saying...

MacUser

"Symantec C++ 8.0 for Power Macintosh is an excellent development for creating Power Mac applications. Its compilers are fast, and its well designed project manager interface is a boon for managing large projects"

MacTech

"All in all the product is incredibly good. I think that maybe Symantec has made life too easy for me."

1995 Byte Magazine
Readers Choice Award
Best Programming Language
Macintosh

Macworld

"Symantec's product is a more polished development environment and has the advantage with its editor, class browser and compiled code."

UNIFORM RESOURCE LOCATORS

By Jim Straus, URLs@mactech.com

Spare your fingers and find the full list online at:
http://www.mactech.com/URLs.html

Welcome to the latest update on Internet information. Java is the hot topic of the moment; everyone is jumping on the bandwagon. A lot of people sure are getting excited about this dynamic object-oriented language with a class library. Java's popularity has focused our attentions away from similar environments, like SmallTalk, Dylan, or NewtonScript, but if the Web is what it takes to move people toward the future of programming, that is great. Now on to the sites.

LATEST UPDATES

Internet Related

How Do I...	http://www.digitalfocus.com/digitalfocus/faq/howdoi.html
Java SDK	http://java.sun.com/new.html
MacJava	http://www.seas.upenn.edu/~mcrae/projects/macjava/
OpenDoc Java	http://summary.net/~breck/java-opendoc.html

New Technologies

ATT Bell Labs	http://www.research.att.com/
DayStar MP Developers	http://www.daystar.com/developer/dev-pageone.html
MacLinux	http://www.mklinux.apple.com
OpenDoc/Bento/SOM	http://www.cilabs.org/

Other Programmer Resources

Pascal Central	http://www.webcom.com/icog/polymorphic/pascal.html

Vendors, Products and Miscellaneous

Always Apple	http://always.apple.com
BareBones SW (BBEdit)	http://www.barebones.com/
Fog City Software	http://www.fogcity.com/
guideWorks	http://www.guideworks.com
NeoLogic	http://www.neologic.com/
Nisus	http://www.nisus-soft.com/
PowerTap	http://members.aol.com/powertap
PPC Fragments	http://www.triplesoft.com/fragment/

INTERNET

Finally, there are some Java development tools appearing for Macintosh. JavaSoft (Sun's newly created division that is supporting Java) has released the first beta version of the Macintosh Java SDK. By the time you read this, it should be much farther along. Natural Intelligence is releasing updates to their Roaster development environment, and Metrowerks and Symantec should have their development environments out. Also, people are thinking about Java and OpenDoc and just general Java info. Java is one of those products that you should keep track of, as it could have a significant impact on the industry. Just as an example of its perceived importance, Bell Labs has pulled some of the biggest names (such as Dennis Ritchie) off of Plan 9, their next-generation OS, and put them onto their own answer to Java (no name as of yet).

Java SDK http://java.sun.com/new.html
Metrowerks http://www.metrowerks.com/
Symantec http://www.symantec.com/
How Do I... http://www.digitalfocus.com/digitalfocus/faq/howdoi.html
MacJava http://www.seas.upenn.edu/~mcrae/projects/macjava/
OpenDoc Java http://summary.net/~breck/java-opendoc.html
ATT Bell Labs http://www.research.att.com/

Sometime during the shakeup at Apple this winter, Apple decided to officially support the Open Software Foundation's porting Linux to the PowerPC. They are pushing hard to make the PowerPC Macintosh into an Internet server, in every possible way that people may want. This Unix variant joins the ranks of Tenon's MachTen, IBM's AIX, and Apple's own AU/X.

MacLinux http://www.mklinux.apple.com
Tenon - MachTen http://www.tenon.com/

MACINTOSH

Now that Gil Amelio is at Apple, he's announced that one of his goals is to provide more visibility into what is going on inside Apple. We will have to see if this works, but in the meantime, there is a new Web site where he has posted some information on what's been done to date. Hopefully, developers and the public will see some of what goes on inside Apple, and will gain confidence that Apple will be a great place in the future. We could all use the good news.

Always Apple http://always.apple.com

I am always interested in how fast a machine can go. There's a limit to how fast a single processor can go (even a PowerPC), and multi-processing is how we will break through that barrier. DayStar has a page with technical information on their multi-processor system. If you are looking for computing power, four 604s will certainly go fast. Now to find a compiler that will use them all!

DayStar MP Developers
http://www.daystar.com/developer/dev-pageone.html

If you need more than the computing power of one machine, how about ganging them up across a net? PowerTap has a library and utilities to allow your programs to use the CPUs on other Macs on a network. It will even use the multiple CPUs in an MP machine. Just think of how much computing your program could get done at night in a typical office. Think of how much spare computing power there is, at night, inside Apple!

PowerTap **http://members.aol.com/powertap**

Neat Non-Macintosh Site of the Month

On the Web, you can find neat maps of just about anything. Xerox PARC put out one of the first maps available to the public on the Internet. You can zoom in on any place in the world. MapQuest also allows you to zoom around, or you can enter an address and see just where it is located. Finally, the Census Bureau has a mapping program that has all sorts of interesting overlays (from all the data they've collected) that can be added to their geographic maps.

MapQuest **http://www.mapquest.com/**
Map Viewer **http://pubweb.parc.xerox.com/map/**
Tiger Mapping Service **http://tiger.census.gov/**

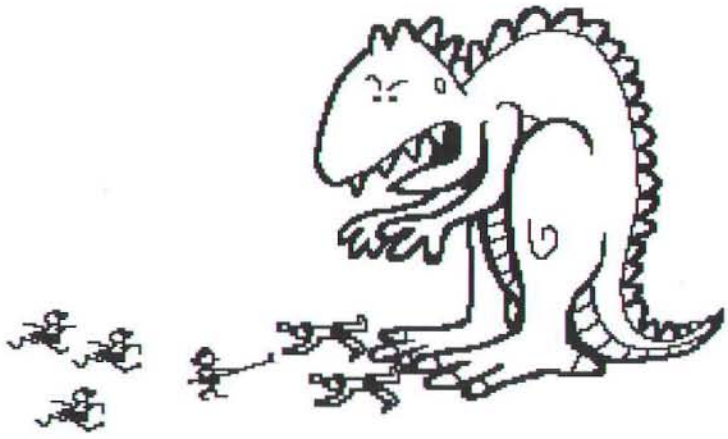
Well, that is it for this month. As always, if you find something interesting, or have updates, send them to URLs@MacTech.com.

Thanks this month to Gil Amelio, Liam Breck, Bill Catambay, Guy Kawasaki, Steve Kortze, Bruce Lawton, Will Mayall, Matt Neuburg, John Powers, Rich Siegel, Joe Zobkiw, and many others for their contributions for their suggestions and pointers to new and old sites.

To receive information on any products advertised in this issue, send your request via Internet: productinfo@xplain.com

Visit MacTech Magazine's Web site!
http://www.mactech.com

Are your projects eating you alive?



Are your projects eating you alive? When time is of the essence the quality of the people and their work means everything...

Swift performs above the standard level. We understand that experience, timing and costs can mean the difference for a successful project. When you need a person or a team to make things happen, call us.

We have provided solutions from the firmware level up. We work with the latest technologies such as Oracle Power Objects, OpenDoc/ODF, & Java to bring the latest options to solving your problems. Our technical services include Project Management, R & D, Training and Technical Support.

Swift can make the difference.
To see how, call (408) 338-1464.

S W I F T
CONSULTING
610 West Dr.
Boulder Creek, CA 95006

SYMANTEC TOP 10

By Craig Conner and Scott Morison

SYMANTEC.™

This monthly column, written by Symantec's Technical Support Engineers, is intended to give our readers technical information on using Symantec products.

With the release of 8.1, we thought we would run through some update questions for both Pascal and C/C++ PPC projects, and then look at some interesting features. We end with some 68K questions and a C++ language update.

Q: With the introduction of the Pascal drop-in for SPM, what is involved in porting Think Pascal code to the PPC?

A: First, we would like to thank Language Systems for creating the drop-in PPC Pascal compiler for SPM. Now, on to the answer. To port code involves several steps. First, you need to update your code following Apple's recommended steps to migrate from 68K code to PowerPC code. Consult Chapter 3 of the `Object Pascal.pdf` file for more specific info. To ease this transition you can use the Universal Headers 2.1 that we released for Think Pascal and do the changes within the Think environment.

Second, when you create the project, you need to include the following libraries, or use the Project Model:

```
InterfaceLib
MathLib
PascalUPPGlue.o
PasLibx.o
PPCANSI.o
PPCRuntime.o
PPCunix.o
```

Third, you will need to add the relevant units into a `USES` statement both in your source code and in the project. (You can also use the `{$I}` syntax used by MPW to include files.)

Q: Can I use block comments in Language Systems Pascal?

A: Yes! LS Pascal is similar to MPW Pascal in that regard. Use braces, like this:

```
{
This is a
block comment
}
```

Q: I'm trying to update a project from 8.0.3 to 8.1, and I get the error message:

```
File "InterfaceLib.xcoff"
Error:   the file was not found.
```

I also get the same message for the `MathLib.xcoff` and `ObjectSupportLib.xcoff`. Why is this?

A: You will need to replace `InterfaceLib.xcoff` with `InterfaceLib`, `MathLib.xcoff` with `MathLib`, and `ObjectSupportLib.xcoff` with `ObjectSupportLib`. These libraries have changed from xcoff library stubs to actual shared library stubs, so these files have lost the `.xcoff` extension. Remove the old libraries and add the new versions.

Q: Okay, I now have the right libraries, but I get a continuous stream of errors like this:

```
Prefix "PowerPC C++": Line 1: While
compiling "CStream_myContents.cpp"
Error:    different configuration for
precompiled header
```

A: The default precompiled header has the Exception Handling switch on, but most converted files convert with the switch off. You can find the switch by choosing **Options...** from the **Project** menu and looking at the PowerPC C++ Language Settings. If that does not help it is time to re-

precompile the header. To quickly precompile the header you can simply choose the **Precompile Header** script from the **Scripts** menu.

Q: Why do I now notice a pause and then a dialog box opening when entering variables in the data window?

A: SC++ 8.1 allows for deferring debugging file generation. This both speeds up compile times and limits the size of the compiled project. The downside is that when you ask for information from a file that doesn't have debug information created yet, the debugger needs to take a moment and create it.

Q: I am using the SPM and trying to get my 68K application to link. ToolServer starts to open, and then it posts a dialog saying `StdCLib could not be found`. Why?

A: ToolServer 3.4 requires the `StdCLibInit` extension to be loaded in your Extensions folder. You can copy it from the 8.1 disk `System Additions` folder.

Q: The Apple Guide tutorials don't show up in my **Help** menu. Why not?

A: For the SPM to load the Apple Guides, you need to have the AppleGuide extension loaded in your Extensions folder. Look for more Apple Guides in the future.

Q: My TPM project complains about not being able to find the ANSI library; what has changed?

A: The following TPM libraries' names have been changed in a minor but important way:

```
ANSI-A4
ANSI-A4++
ANSI-small
ANSI-small++
CPlusLib-A4
```

These libraries all used to use em-dashes (option-hyphen) in their names instead of hyphens. Due to problems this causes on Japanese systems, the em-dashes have been changed to regular hyphens. Projects which use these libraries will have to remove and re-add them in order for TPM to be able to recognize the libraries with their new names.

Q: How do I use `AppendDITL()` in a Think Pascal project?

A: This call was new to System 7 and never really made it into the old toolbox library. It is, however, declared in the `CommToolBox.lib`. Here is the way to get it working. Add `CommToolBox.lib` to your project.

Create a unit (or modify an existing one) that declares `AppendDITL` as external. Something like:

```
unit myDeclarations;
interface

procedure AppendDITL(
  theDialog:DialogPtr;
  theHandle:Handle;
  method:integer);

implementation

procedure AppendDITL(
  theDialog:DialogPtr;
  theHandle:Handle;
  method:integer);
        external;

end.
```

Notice that I changed the method parameter to an integer type, so that I would not have to declare `DITLMethod` also. (You may want to declare that, and the method constants also for elegance.)

Q: Do I still have to use the pragmas for instantiation of static templates?

A: No. The compiler now accepts template-explicit instantiation as outlined in the ANSI C++ draft standard (dated 9/26/95) Section 14.4 pp. 14-15. The following pre-8.1.0 statements:

```
template <class T> void f(T t);
template <class T> class X { };

#pragma template access public
#pragma template f(int)
#pragma template X<int>
```

are equivalent to:

```
template <class T> void f(T t);
template <class T> class X { };

template void f(int);
template class X<int>;
```

Note: The old method of using `#pragma template` directives continues to be supported.

Special thanks to: Glenn Austin, Michael Hopkins, Kevin Quah, and Mark Baldwin.

By Mike Cohen, Casper, WY

Writing CGI Applications with 4D

Beam your web pages into the 4th DIMENSION

In previous articles [*in, for example, MacTech Magazine 11.7, 11.8, 11.9, 11.12, and 12.1 – man*], you were shown how WebSTAR or MacHTTP can interface with CGI applications. In this article, I'll demonstrate how to use this same CGI interface to have your Web server communicate with 4th DIMENSION, using System 7 Pack from ISIS International.

About System 7 Pack

System 7 Pack, from ISIS International, is a 4th DIMENSION external package that lets you send and receive Apple events in your databases. With System 7 Pack, 4D applications can control other applications, such as Microsoft Excel or QuarkXPress, and can be accessed by other local or remote applications. Once it possesses an Apple event handler, your 4D database can act as a CGI application for WebSTAR or MacHTTP.

CGI Applications

If you've read previous articles in this series, you probably know the basics of CGI applications, but I'll review them here. When a Web client requests a file, WebSTAR (or MacHTTP) uses the filename's suffix to determine how the file should be handled. In most cases, the file will simply be returned to the client. However, for a file type of `.cgi` or `.acgi`, the application will be launched, if it isn't already running, and an Apple event will be sent to it with information passed from the client. After processing that event, the CGI application returns HTML text to WebSTAR.

Using 4D to Write CGI Applications

The first thing you need to do is to put a copy or alias of either 4D itself or a compiled 4D database merged with Runtime into your WebSTAR folder and give it a name ending in `.cgi` or `.acgi`. If you use the suffix `.acgi`, your application will be called asynchronously: WebSTAR won't wait for an Apple event to complete before sending additional requests. Such an application must be prepared to have multiple outstanding requests, and may need to use semaphores to prevent simultaneous access to global variables or data files.

Install an Apple Event Handler

The first thing your CGI application must do when 4D starts up is to install a handler for the Apple event WebSTAR sends you. The following line of code in our startup procedure will take care of that:

```
$Err := HandleAEVT ("WWW Ω";"sdoc";"CGI Handler")
```

This System 7 Pack function tells 4D to call the procedure named `CGI Handler` whenever it receives an Apple event of class `'WWW Ω'` and ID `'sdoc'`, which is the event that WebSTAR sends to CGI applications. The full startup procedure (see Listing 1) also initializes several global variables that the rest of the application will use.

Since many of our global variables will be used by the Apple event handlers, which run in a separate process, we use

Mike Cohen is the chief programmer and one of the founders of ISIS International, a company that develops business productivity and connectivity software for the Mac and other PCs and PDAs. He programs in C, C++, 4D, and NewtonScript among other languages. To find his office, listen for the sound of West African music.
Jon Wiederspan was Contributing Editor for this article.

interprocess variables identified by a name beginning with the diamond character (◊). The most important variables, ◊ae_headRslt, ◊ae_TextHdr, and ◊ae_PictHdr, contain standard HTTP headers that must be returned along with any data we send back to WebSTAR.

When 4D receives an Apple event, it automatically starts a process called Apple Event Manager, in which all event handler procedures run. With some versions of 4D (most notably 4D Server 1.0.5 and any non-server version earlier than 3.0.5), the first Apple event received is discarded or takes a long time to handle. To avoid these problems, we have our startup procedure send a dummy Apple event to itself.

Listing 1: STARTUP

```
` let's define a few global variables here
◊crlf:=Char(13)+Char(10)
◊q:=Char(34)   ` double quote
◊nl:="<br>"    ` html line break

` standard HTML header
◊ae_headRslt:=
   "HTTP/1.0 200 OK"+◊crlf+"Server: WebSTAR"+◊crlf
◊ae_headRslt:=
   ◊ae_headRslt+"MIME-Version: 1.0"+◊crlf
◊ae_TextHdr:=
   ◊ae_headRslt+"Content-type: text/html"+◊crlf+◊crlf
◊ae_PictHdr:=
   ◊ae_headRslt+"Content-type:image/gif"+◊crlf+◊crlf

` install our Apple event handlers
$Err:=HandleAEVT ("WWW0":"sdoc";"CGI Handler")

` some versions of 4D lose the first Apple event or take unusually long
` to process it, so we'll send ourselves a bogus event to start the
` AppleEvent Manager process which will handle all incoming events
$Err:=MakeAddress("4D05";$myself)
$Err:=SendAEVT($myself;"XXXX";"XXXX";"-")
$Err:=DisposeDesc($myself)
```

Hand Off the Apple Event

Next, you need to write a procedure that will be executed in response to the Apple event. It's important for this procedure to run as quickly as possible, since each incoming Apple event is queued and dispatched to this single process. A good approach is to simply call AESuspend (a function introduced in System 7 Pack 3.8.3) and pass the event returned from it to another process. For the sake of simplicity, I merely start a new process here, but for better performance, you should start the process when the application starts up, suspend it, and wake it up each time an event comes in.

Listing 2: CGI Handler

```
$Err := AESuspend (◊theEvent;◊theReply)
New process("CGI Script";32000;"CGI Script Handler")
```

The function AESuspend makes a copy of the incoming event and reply, and informs the AppleEvent Manager that it shouldn't (as it normally does) automatically send the reply when the procedure finishes. Note that you must use interprocess variables (indicated by a variable name that begins with ◊), since the event will later be handled in a separate process.

The New Process command then starts up a separate process called CGI Script Handler that will run the

procedure `CGI Script` to do the actual event handling. Once this procedure finishes, 4D will then be free to receive additional Apple events while the `CGI Script Handler` process finishes handling the event.

Process the Apple Event

Finally, the function `CGI Script` (see Listing 3), which will be run in a separate process, finishes handling the request and sends the reply to WebSTAR. The first thing we do is to save the Apple event and reply in process variables, since a new Apple event coming in at this time would modify the interprocess variables. Next, we initialize several variables we use to extract data from the incoming Apple event. We then extract the path argument, sent as the direct object using the System 7 Pack function `GetTextParam`. The additional arguments sent by WebSTAR, including the method, search, and post arguments, are also extracted by calling `GetTextParam`. Next, we inspect the various arguments to decide how to handle the request. In this procedure, we handle all `POST` requests by adding a record to the database.When we receive a `GET` request, if no `PATH` argument is given, we simply return a list of all records containing the search string. If the `PATH` argument begins with `rec`, we assume the search argument is a record number and return the full record. If the `PATH` argument begins with `pict`, we use the search argument as a record number and return a picture field from that record as a GIF image.

Listing 3: CGI Script

```
evt:=◊theEvent
rep:=◊theReply

` assume the reply format will be TEXT
$hdr:=◊ae_TextHdr

C_TEXT(http_meth)    `the method - either "GET" or "POST"
C_TEXT(http_srcha)   `the search arguments
C_TEXT(http_post)    `the post argument
C_TEXT(http_path)    `the path argument
C_TEXT(use_address)  `the user issuing this request

$err:=GetTextParam (evt;"----";http_path)
   `extracts the "direct object" parameter of the
   `Apple event which caused this procedure to be launched
   `the "direct" object of an Apple event passed from WebSTAR is the
   `"path" argument

   `get other pieces from this apple event
$err:=GetTextParam (evt;"meth";http_meth)
$err:=GetTextParam (evt;"kfor";http_srcha)
$err:=GetTextParam (evt;"post";http_post)
$err:=GetTextParam (evt;"addr";use_email)

Case of
   : (http_meth="post")   ` handle a post request
      $ostr:=DoPost (http_post)
   : ((http_srcha="") & (http_path=""))   ` all blank
      $ostr:="No search terms were given."+
           "What do I search for?"+◊crlf
   : http_path="rec@")    ` handle a record # detail request
      $ostr:=GetRecord (http_srcha)
   : (http_path="pict@")  ` handle an image request
      $ostr:=GetImage (http_srcha)
      $hdr:=◊ae_PictHdr
   : (http_path="")       `it's a simple search
      $ostr:=DoSearch (http_srcha)

   Else
      $ostr:="ERROR- The path '"+http_path+
```

```
        "' is not supported."+0crlf
End case

` now, send the reply back to WebSTAR for this Apple event
$err:=PutTextParam (rep;"----";$hdr+$ostr)

` finish processing this event and send the reply
$err:=AEResume (evt;rep)
```

Let's take a closer look at this procedure.

The first two lines copy the event and reply from interprocess variables into process variables, both for convenience and to avoid trouble if they get changed by another process. If you expect to have many incoming events, you should probably use an array into which the Apple event handler pushes values, and use this function to pop them out.

We then use GetTextParam to extract the parameters that WebSTAR passed to the CGI script. The direct object is the path value (the text that follows the $ in your HTML request). The other important values are the method (http_meth, or the 'meth' descriptor in the Apple event), which is either GET or POST, the post argument (http_post, or the 'post' descriptor in the Apple event), which contains the data entered in the form which uses a POST method, and the search argument (http_srcha, or the 'kfor' descriptor in the Apple event), which contains the text that follows the ? in your HTML request.

Next, we use a case to determine what type of request we got and take the appropriate action for it. If we got a POST request, we call the function DoPost, which parses the post arguments and updates the database with the new data. The other requests we handle are: a simple GET request with no path value, in which case we simply do a search and return a list of matching records; a GET request with a path value of REC, for which we return the full details of a record specified by number; and a GET request with a path value of PICT, for which we return a picture field from a record specifed by number. Note that string comparisons are case insensitive and can include 4D's wildcard character (@).

Finally, we call PutTextParam to place the text to be returned into the reply event, and call the function AEResume, which informs the AppleEvent Manager that we're finished handling the event that we previously suspended and sends the reply event back to WebSTAR.

Process POST Data

The procedure DoPost handles a POST request. In this case, we simply add a record to the database. Your system should probably check to see if the record already exists before it adds a new record.

The POST arguments, which provide the data to be saved in the record, are sent as a string of field names and values, such as lname=Mike&fname=Cohen&city=Casper&state=WY. Spaces are replaced with plus signs [*only for Netscape and Mosaic; other browsers replace spaces with* '%20' *– jaw*], and other special characters are encoded as hexadecimal values. You could write a procedure to parse the string in 4D, but it

would be very slow. The easiest way to handle the POST string is to use the Parse Post Args function of Wayne K. Walrath's excellent Acme Script Widgets OSAX.

AppleScript OSAXen are additions to the AppleScript language that work by installing a system-level event handler. You can call any scripting addition by sending the appropriate Apple event either to your own application or to the system, without having to call AppleScript. In this case, I send an event to the system, since some versions of 4D seem to have problems with re-entrancy when sending an Apple event to themselves.

Listing 4: DoPost

```
` create a new record
CREATE RECORD([Names])

` call the parse post args function of Acme Script Widgets
$err:=MakeAddress ("MACS";$system)
$err:=CreateAEVT ("zCGI";"ppar";$system;$aevt)
$err:=PutTextParam ($aevt;"----";$1)
$err:=SendAppleEvent ($aevt;$reply;kAEWaitReply ;-1)
$err:=GetList ($reply;"----";MyArray)

` examine the array of fields and values to determine
` where to place each piece of data
For ($i;1;Size of array(myArray))
  $err:=GetNthItem (MyArray{$i};1;$aKey;$aType;$theField)
  $err:=GetNthItem (MyArray{$i};2;$aKey;$aType;$theValue)
  Case of
    : ($theField="fname")
      [Names]First Name:=$theValue
    : ($theField="lname")
      [Names]Last Name:=$theValue
    : ($theField="address")
      [Names]Address 1:=$theValue
    : ($theField="city")
      [Names]City:=$theValue
    : ($theField="state")
      [Names]State:=$theValue
    : ($theField="zip")
      [Names]Zip:=$theValue
    : ($theField="country")
      [Names]Country:=$theValue
    : ($theField="email")
      [Names]Email:=$theValue
  End case
End for

` clean up everything we allocated
$err:=DisposeDesc ($reply)
$err:=DisposeDesc ($aevt)
$err:=DisposeDesc ($system)

SAVE RECORD([Names])
UNLOAD RECORD([Names])
$0:="Record Added."
```

The first thing we do with this code is to create a new record. Next, we parse the POST arguments passed to our script by calling the ACME Script Widgets OSAX. The function MakeAddress is used to create a target address referring to the system process. We use that target address to create an Apple event of class 'zCGI' and ID 'ppar', which invokes the Parse Post Args function. After we send the event, we extract the array of field name and value pairs it returns.

Next, we examine each name/value pair and use the data to update the record. The array we got back from `Parse Post Args` consists of a series of sub-arrays containing a field name and value. We inspect each field name and use it to determine which data field the value should be placed into.

After we finish processing the data, we dispose of all the Apple event records and other descriptors we created, to avoid any memory leaks.

Finally, we save and unload the new record. Our procedure returns the text "Record Added". This will be sent back to WebSTAR and displayed in the browser as an acknowledgement that the form was accepted.

Process SEARCH Data

The procedure `DoSearch` handles a simple search request by returning a formatted list of all records which match the search string. Each item is returned as a URL that will display the entire record if it is clicked. This procedure will be passed the search argument and will use it to do a substring search. If any records match the search string (in this case, we use the last name field), they will be returned as a formatted list of first and last name. Clicking one of them in your browser will display the full details for that record.

Listing 5: DoSearch

```
` our search form has a single unnamed field, so we strip off the
` equal sign for the label that precedes the search value
If ($1≤12="=")
   $1:=Substring($1;2)
End if
SEARCH([Names];[Names]Last Name=$1)
If (Records in selection([Names])=0)
   $reply:="<strong>No matching items.</strong>"
Else
   $reply:="<Title>Search Results</Title>"+
      "<b>Matching items:</b>"+◊crlf+"<ul>"+◊crlf
   FIRST RECORD([Names])
   For ($i;1;Records in selection([Names]))
      $fn:=[Names]First Name
      $ln:=[Names]Last Name
      $reply:=$reply+
           "<li> <A HREF="+◊q+"4d.acgi$recno?"+
           String(Record number([Names]))
      $reply:=$reply+◊q+">"+$fn+" "+$ln+"</A>"+◊crlf
      UNLOAD RECORD([Names])
      NEXT RECORD([Names])
   End for
   $reply:=$reply+"</ul> "
End if
$0:=$reply
```

The search result is returned as HTML text. If any matching records are found, we format them as a list. Each item is a clickable link that will send a new request back to our application, which will display the full details of our record. The URLs will look something like this:

```
<li><A HREF="4d.acgi$recno?1">Mike Cohen</A>
```

Retrieving Detailed Records

The next procedure, `GetRecord`, is called from the main Apple event handler in response to a record number detail request. It takes a record number passed as the search argument, goes directly to that record, and returns a formatted name and address label for that record.

Listing 6: GetRecord

```
$fn:=Num($1)
DEFAULT FILE([Names])
GOTO RECORD($fn)
$reply:="<Title>Search Results</Title>"+
    "<STRONG>Client Info:</STRONG><p>"+0crlf
If ([Names]showImage)
  $reply:=$reply+
    "<img src="+0q+"4d.acgi$pict?"+$1+0q+">"+0crlf
End if
$reply:=$reply+[Names]First Name+" "+[Names]Last Name+0nl
$reply:=$reply+[Names]Title+0nl+[Names]Company+0nl
$reply:=$reply+[Names]Address 1+0nl+[Names]Address 2+0nl
$reply:=$reply+[Names]City+", "+[Names]State+
    "  "+[Names]Zip+0nl
$reply:=$reply+[Names]Phone 1+0nl+[Names]Phone 2+0nl
If ([Names]Email#"")
  $reply:=$reply+"<A HREF=mailto:"+
    [Names]Email+">"+[Names]Email+"</A>"+0crlf
End if
UNLOAD RECORD([Names])
$0:=$reply
```

This procedure will go directly to the specified record and return the data formatted as HTML text. If an image is present, it will be passed as another ACGI request that will be sent back to our application to display that image. If an email address is available, it will be returned as a clickable `mailto` URL.

Handling Requests for Images

Images in a 4D database are stored as PICT fields. Since PICT isn't supported by most Web browsers, we must convert it to GIF format before we return it to WebSTAR. The procedure `GetImage` will return a PICT field from the specified record as a GIF image to be displayed, using Yves Piguet's freeware Clip2Gif utility. As in the previous procedure, we use the search string as a record number and go directly to that record.

Listing 7: GetImage

```
$fn:=Num($1)
DEFAULT FILE([Names])
GOTO RECORD($fn)
$gifText:=""
` if Clip2Gif isn't running, launch it now
If (IsRunning ("c2gf")=0)
  $err:=Launch ("c2gf":"")
End if
` send a "save" event to clip2gif to convert PICT to GIF
$err:=MakeAddress ("c2gf";$Clip2Gif)
If ($err=0)
  $err:=CreateAEVT ("core":"save";$Clip2Gif;$aevt)
  If ($err=0)
    $err:=PutPicParam ($aevt;"----";1;[Names]Logo)
    $err:=PutLongParam ($aevt;"fltp":"type":Long ("GIFf"))
    $err:=PutLongParam ($aevt;"kfil":"type":Long ("itxt"))
    $err:=SendAppleEvent ($aevt;$reply;kAEWaitReply :-2)
    If ($err=0)
      $err:=GetTextParam ($reply;"----";$gifText)
    End if
    $err:=DisposeDesc ($reply)
    $err:=DisposeDesc ($aevt)
  End if
  $err:=DisposeDesc ($Clip2Gif)
End if
UNLOAD RECORD([Names])
$0:=$gifText
```

Again, let's review. First, we make sure Clip2Gif is running, and if it isn't, we launch it. Next, we send a `'save'` event with the PICT data and ask to have it returned in GIF format. Clip2GIF has several options for creating GIF images, including transparency and interlacing. (Information about these options and how to access them via Apple events is included with the

software.) We then extract the GIF data from the reply and return it as the result of this procedure, to be sent back to WebSTAR.

Sample HTML Forms

Now that we have the code for our CGI application, here are some sample HTML forms that can be used to drive it. The `DBSearch.html` form (Listing 8) will start a search of the database and return a list of links to matching records.

Listing 8: DBSearch.html

```
<HTML>
<HEAD><TITLE>Search Database</TITLE></HEAD>
<BODY>
<FORM ACTION="4d.acgi" METHOD=GET>
Item to search for:<INPUT name="" TYPE="text" SIZE=32>
(use '@' as a wildcard)
<p>
<INPUT TYPE="submit" VALUE="Search">
<INPUT TYPE="reset" VALUE="Cancel">
</FORM>
</BODY>
</HTML>
```

The `DBUpdate.html` form (Listing 9) will send a `POST` request to create a new record in the database.

Listing 9: DBUpdate.html

```
<HTML>
<HEAD><TITLE>Update Database Entry</TITLE></HEAD>
<BODY>
<FORM ACTION="4d.acgi" METHOD=POST>
Name:<INPUT TYPE="text" NAME="fname" SIZE=20>
<INPUT TYPE="text" NAME="lname" SIZE=20><P>
Address:<INPUT TYPE="text" NAME="address" SIZE=40><P>
City:<INPUT TYPE="text" NAME="city" SIZE=15>
State:<INPUT TYPE="text" NAME="state" SIZE=2>
Zip:<INPUT TYPE="text" NAME="zip" SIZE=10><p>
Country:<INPUT TYPE="text" NAME="country" SIZE=15><p>
Email:<INPUT TYPE="text" NAME="email" SIZE=20><p>
<SELECT NAME="System Type">
<OPTION>Macintosh II series
<OPTION>Macintosh Quadra
<OPTION>PowerMac
<OPTION>Powerbook or Duo
<OPTION>PC
<OPTION>Other
</SELECT><P>
<INPUT TYPE="submit" VALUE="Update">
<INPUT TYPE="reset" VALUE="Cancel">
</FORM>
</BODY>
</HTML>
```

Conclusion

For most Web sites, the most important function of the server is to deliver information to users. If you already have data in a 4D database, you can now easily make this information available directly from your Web pages. You can also develop Web-based front ends to your 4D applications for uses such as kiosks, where the 4D user interface is inappropriate or too complex. Once you get started I am sure that you will find dozens of ways that you can use a 4D CGI application to improve your Web site.

NEW APPLE TECHNOLOGY

By Andrew Nemeth

Panoramic Reality

Getting started with QuickTime VR

Introduction

Apple's QuickTime VR technology is possibly one of the coolest things to happen to digital imaging in the last five years. With it, you can produce and view 360-degree panoramas of real or computer-generated scenes, without the need for expensive panoramic cameras or any other fancy equipment usually associated with VR.

You create a panorama by photographing a scene with a standard 35mm camera atop a tripod, taking twelve to eighteen photographs to capture a full 360-degree view. Then, using Apple's software, you "stitch" these images into a single PICT file, which is then processed further to create the final, user-navigable movie, playable on both Macintosh and WINTEL personal computers.

As there are already a number of general introductory articles on QuickTime VR (see the references at the end of this article), the following will concentrate instead on the tools and techniques used to create a photographic panorama.

Photographic Equipment

To keep costs down, and for the sake of portability, use of the following is suggested:

- 35mm SLR camera
- 24mm f2.8 lens
- (Lots of) 400 ISO low-contrast C41 negative film
- Sturdy tripod with built-in spirit levels
- Kaidan QP-1A QuickPan Panoramic Base
- Vertical camera mounting bracket

Note that some of the above is different to that recommended by Apple in their QuickTime VR 1.0 Authoring Tools. The reasons for these differences will be noted below.

35mm camera

I use two cameras for my work: a 25-year-old Nikon F (pictured in Figure 4) and a more recent Nikon F90x. These are used depending on the locality, with the F90x for general use and the older – indestructible – F reserved for more rugged activity.

Although not absolutely necessary, a 35mm SLR camera gives you the option of using different focal length lenses on the same camera body, meaning you can use wider lenses indoors and longer ones for outside work. They also tend to be better built and feature more accurate focusing and metering. A 35mm camera also gives you a much higher quality of image than the current crop of digital cameras.

As regards light meters, I prefer to use a hand-held model. Again, this is not essential, but I find it to be more convenient than the built-in meters found in cameras (especially the Nikon F, which does not have *any* kind of meter!).

24mm lens

As the first of many departures from Apple's suggestions, a relatively inexpensive Sigma f2.8 24mm lens is favoured for

Andrew Nemeth must be the only Mac developer who does not own a cat. He lives in Warrimoo, Australia, and can be reached at aznemeng@zeta.org.au. His WWW page is located at http://www.zeta.org.au/~aznemeng.

outdoors work rather than the Apple-sanctioned Nikkor f3.5 15mm. Although the wider "Apple" lens does have greater vertical and horizontal coverage (allowing for a reduction in the number of individual photographs required in order to capture a scene), there are two major problems with it: firstly, at a list price of US$2000 the 15mm is *outrageously* expensive; and secondly, its field of view is *too* wide for outdoors work, with everything more than 3 meters away looking as though it was on the edge of the observable horizon.

Although other wide-angle lenses could be used (18, 20, 28, 35mm), the 24mm is a good compromise on a cost/image-width basis. The down-side is that because Apple has assumed everyone will be doing QuickTime VR with a 15mm lens, adjustments have to be made to the MPW scripts which come with the Authoring Tools in order to render the images properly (more on this further down).

400 ISO low-contrast film

A faster film gives you more scope to capture scenes whatever the prevailing light. Although a slower film (50 or 100 ISO) is sharper and has less grain, for QuickTime VR such advantages are lost during the software processing. Furthermore, slower film is *too* slow to use indoors.

A low-contrast C41 film is recommended because it is the best kind of colour film for capturing all the levels of brightness in a scene. A film like Fuji's NPH 400 is sharp, fine-grained and – as it is designed for professional flash-lit photography – more than capable of dealing with harsh outdoor light.

By the way, be warned that QuickTime VR *chews* through film – at 16 images per panorama you can fit only two scenes onto a roll.

Figure 1. Tripod head with double spirit levels

Tripod and head

The sturdier the better. The one pictured (a Manfrotto, see Figure 1) is a 5kg behemoth which features two built-in spirit levels to allow you to quickly level the head prior to attaching the QuickTime VR–related brackets. In case you think you can get away with using a lighter tripod, think again. You really do need a heavy-duty tripod for this kind of work, because you don't want it flapping around in the breeze or moving suddenly in the middle of a sequence of shots. (Okay, I relent: for people with bad backs and deep pockets, the Gitzo model 1228 tripod is a good carbon fibre substitute.)

You also need the spirit levels, because the QuickTime VR software insists on the camera being absolutely level when photographing a scene.

Camera mounting brackets

For the Authoring Tools to work, each 360° scene must be photographed by taking a series of overlapping shots using a vertically oriented camera. To avoid parallax errors when doing this, you must position the optical centre of the lens directly over the axis of rotation (see Figure 4).

For an example of parallax error due to off-centre rotation, hold one finger 10 cm from your face, close one eye, focus onto the background, and then turn your head slowly from side to side – notice how the background appears to shift from side to side behind your hand? Now, keeping your head and hand still, turn your eyes from left to right – this time the background *doesn't* move relative to your finger! In the first case your eyes are not centred on the axis of rotation, and hence the parallax error when you turn your head. In the second example you are turning your eyes, the centres of which *do* coincide with the turning axis!

The set-up I use is shown in Figures 2, 3 and 4. A quick-mount plate has been added onto the Kaidan QP-1A Base (Figure 2) to make it easier to attach to the tripod head. Black gaff-tape has also been stuck over the "deck" of the plate, as its highly reflective silver finish caused lens-flare whenever the camera was rotated over it.

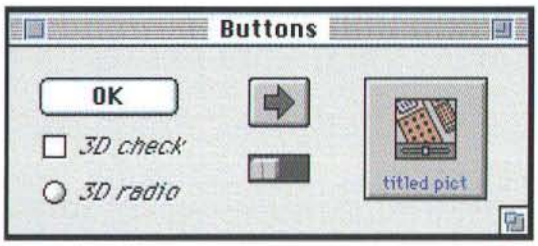
I prefer to use the QP-1A rather than Apple's scheme of mounting a second tripod head, because it is smaller, lighter, and you can set it to "click-stop" at the number of shots you wish to take in a pan. (For the 24mm lens it has been set for 16 detents. It can also be set for 8, 12, 14, or 18.)

Figure 2. Kaidan QP-1A Base with Manfrotto quickmount plate

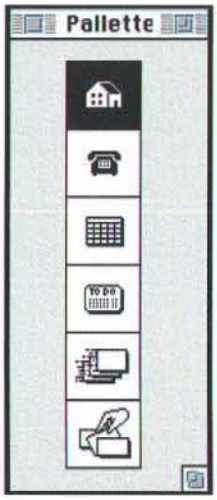
The bracket used to mount the camera onto the QP-1A is something I made myself from quarter-inch steel brackets, bolts and Araldite (Figure 3). This is a more durable and *much* cheaper solution than Kaidan's QPU-1 (US$180) bracket. However, because it is not adjustable (unlike the QPU-1), separate brackets have had to be made for each camera used.

AppMaker

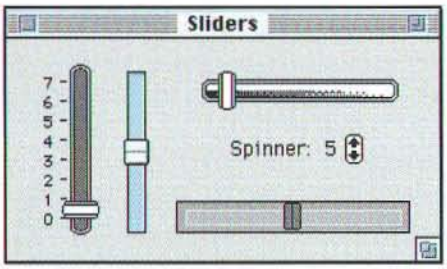
Your Assistant Programmer



AppMaker makes it fast and easy to build a user interface. Just point and click, then let AppMaker create resources and generate source code. AppMaker generates excellent source code for most popular languages and frameworks.



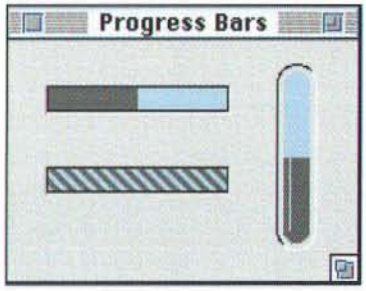
It's like having an expert assistant working for you. Use it as a prototyping and productivity tool. You concentrate on the unique features of your application instead of the standard parts. Novices read the AppMaker code to learn Macintosh programming techniques.



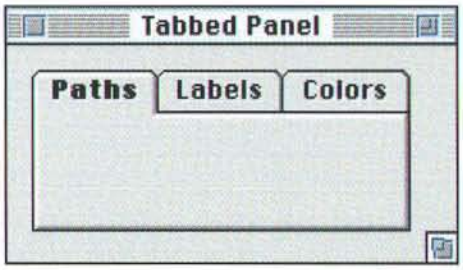
AppMaker now supports a wide variety of gadgets including 3D controls, picture buttons, sliders, progress bars, and tabbed panels. Users can easily customize and extend AppMaker to support other custom CDEFs.

The built-in simulator lets you try out all these new gadgets.

Interested in OpenDoc and the OpenDoc Development Framework (ODF)? AppMaker makes it easy to get started. Just design your interface then generate code to make an OpenDoc part editor.



AppMaker also supports CodeWarrior PowerPlant, procedural C, and PowerMac development. A single AppMaker document can now generate code and resources for a wide variety of languages, frameworks, and eventually, platforms.



Includes one-year automatic update service with at least three CDs per year.

Available from MacTech Mail Order Store for just $299 with a 60-day money-back guarantee.

Demo versions on AOL: keyword Bowers, on Compuserve in the MacDev forum, and at ftp://members.aol.com/bowersdev.

B•O•W•E•R•S
Development

P.O. Box 929, Grantham, NH 03753 • 603-863-0945
bowersdev@aol.com • 70731.3710@compuserve.com

Figure 3. Kaidan QP-1A base and (custom) 35mm bracket

The entire rig is compact, strong and very easy to assemble (Figure 4). Because the vertical "home-brew" bracket has already been pre-aligned (when it was made), the only calibration required when setting up on-site is the leveling of the tripod head.

Figure 4. Full QuickTime VR rig with camera and 24mm lens

Photographing The Scene

With the rig set up and the camera level, you will need to take a series of overlapping, vertically-oriented photographs to cover a scene. The amount of overlap is in the order of 30–50% and is needed by the stitching tool to help it align adjacent images when it pieces together the final panorama. The more overlap the better, but too much overlap equals wasted film and long stitching times.

Figuring out the amount of overlap (and thus how many photographs to take per panorama) is very much an educated guess, with 16 shots for a 24mm lens appearing to be adequate. Some people shoot 18, some live (very) dangerously and try to scrape by with only 12.

There are a number of things to keep in mind when photographing a scene:

- Your camera rig must be (and remain) asolutely dead-level. The more out of level you are, the harder it will be for the stitcher to piece the images together.
- Photograph in a left-to-right direction. The stitching tool assumes this and you will get very weird results otherwise.
- Keep the exposure identical for each shot. This means you will have to meter the entire scene carefully prior to shooting and then take all your exposures using an averaged value. If you alter the exposure for each individual shot then you will get a banding effect when you come to piece all the images together.
- Use a small lens aperture to make sure everything is in focus. A value of f16–22 gives you coverage from 1.5m to infinity.
- Take your shots as fast as you can without knocking your camera out of alignment. You have to be fast because the light and clouds and cars and people have a habit of not staying still.

With 16 exposures per panorama, you can fit two scenes comfortably onto a 36-exposure roll of film, with a few shots left over as spares.

Developing and Digitising the Photographs

Apple advises you to develop/scan your images with Kodak's PhotoCD process – which may be great for people who live in the U.S. but is less than optimal for everybody else. In Australia, for example, the turn-around time for PhotoCD is in the order of 2+ weeks, with the cost averaging A$80 per roll of 36-exposure film. Thus if you are intending to do lots of QuickTime VR, you may be better off investing in a 35mm film scanner and digitising the images yourself!

This is what I and a few others have done. After development of the negatives, they are scanned with a Polaroid SprintScan 35 via Photoshop (16 images require 25 minutes). When the scans are no longer needed, they are archived with Stuffit and 100 MB zip discs.

To justify the purchase of a scanner, you really must be looking at doing more than just a handful of QuickTime VR panoramas; otherwise PhotoCD is the way to go. A tip: try to use the "Portfolio CD" service, as this will allow you to cram five hundred 768×512 images onto a single CD.

Computer Tools

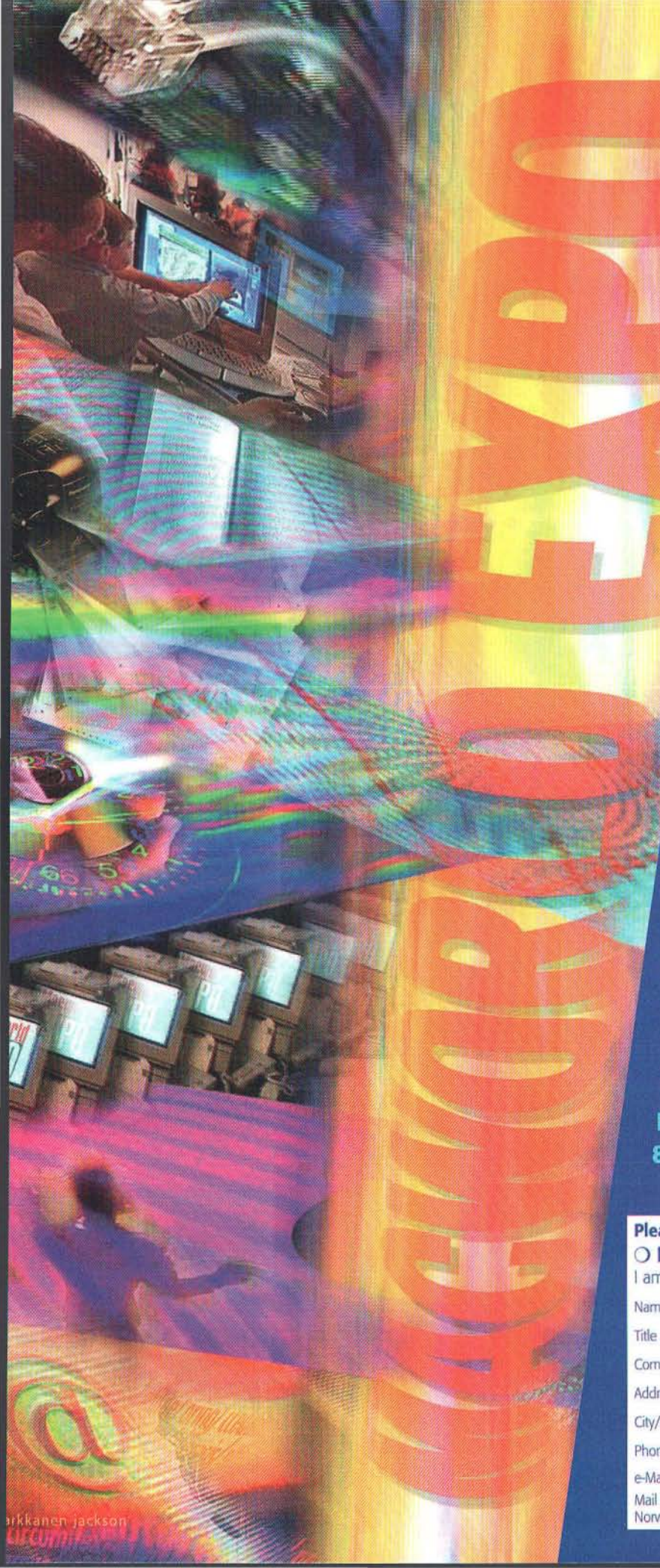
QuickTime VR requires the following:

- Fast PowerPC with 48MB physical RAM and 100MB free disc space
- Colour monitor with millions of colours
- MPW (native 3.4 and beyond)
- Adobe Photoshop 3.0.x
- HyperCard 2.3

As you can see, these are pretty stiff requirements!

PowerPC

Although Apple suggests you can do QuickTime VR development using a 68K computer (68030, 68040), in reality the tools run so slowly that you cannot. For example, it takes 4



It's about business-to-business, home-to-office, artist-to-client, teacher-to-child.

It's about having the knowledge and the know-how to be the best.

It's about making smart choices today and learning about what you need for tomorrow.

It's MACWORLD Expo.

See the best in Macintosh and complimentary tools and technologies. Explore an exhibit floor with everything you've ever wondered about. . . the internet, networking, databases, education, desktop applications, music, design, publishing, illustration, animation, CAD/CAM, programming, entertainment and multimedia. With a complete conference program featuring over 80 sessions, this is your opportunity to interact with leading experts in the industry and get the knowledge and solutions you've been seeking.

MACWORLD Expo/Boston

MACWORLD Expo/San Francisco

See us on the WWW at:
http://www.mha.com/macworldexpo

For more information call
800-645-EXPO x100

Please send me more information on MACWORLD Expo!

❍ **Boston** ❍ **San Francisco**

I am interested in: ❍ Exhibiting ❍ Attending

MT

Name_____

Title _____

Company_____

Address _____

City/State/Zip _____

Phone_____ Fax_____

e-Mail_____

Mail to: MHA Event Management, 1400 Providence Hwy., P.O. Box 9103
Norwood MA 02062. Or Fax to: 617-440-0357 Phone: 617-551-9800

minutes to stitch 16 images on a PowerPC 7500, whereas on a Quadra 800 it requires slightly over an hour. Dicing is similarly glacial on a 68K machine: 25 minutes as opposed to three on a PowerPC.

You need a minimum of 48MB of physical RAM because you will have to allocate at least 40MB of it to MPW. Yes, MPW...

Native MPW

At a time when Macintosh developers are abandoning it, for Apple to release a multi-media development suite dependent on MPW is perverse to say the least – yet this is exactly what they have done with the QuickTime VR 1.0 Authoring Suite.

In order to piece together the images, you use the "stitch" MPW tool. To make a QuickTime VR movie, you first dice (compress) the stitched image with the "p2mv" tool and then make the movie with the "msnm" tool. You drive these and other tools in the time-discredited MPW fashion of executing lines of script in a worksheet. Joy.

APDA offers specials on bundles of QuickTime VR kits with MPW Pro – ignore them. Since you only need to run a small sub-set of MPW (still a bloated 8MB), save your money and use the MPW which comes with the CodeWarrior or Symantec tools.

Adobe Photoshop 3.0

This is *essential* for QuickTime VR development. After stitching you will have to retouch the image, balance the colour, adjust the gamma and a host of other things. It is surely no coincidence that the stitch tool outputs its resultant panorama with a Photoshop creator and file type!

HyperCard 2.3

You will need this only if you intend to produce multi-node movies. This will not be dealt with in this article.

The QuickTime VR Software Process

Figure 5 outlines the procedure required to convert your sequence of scanned images into a single QuickTime VR movie. As you can see, extensive use is made of MPW.

Stitching the images

This is where you take the separate images (see the Sydney Opera House sequence in Figure 6) and – using software – meld them into a seamless whole (Figures 7, 8). In essence what you are doing is using software to produce the same kind of cylindrically distorted image a specialist panoramic camera would create, for approximately one-quarter of the cost.

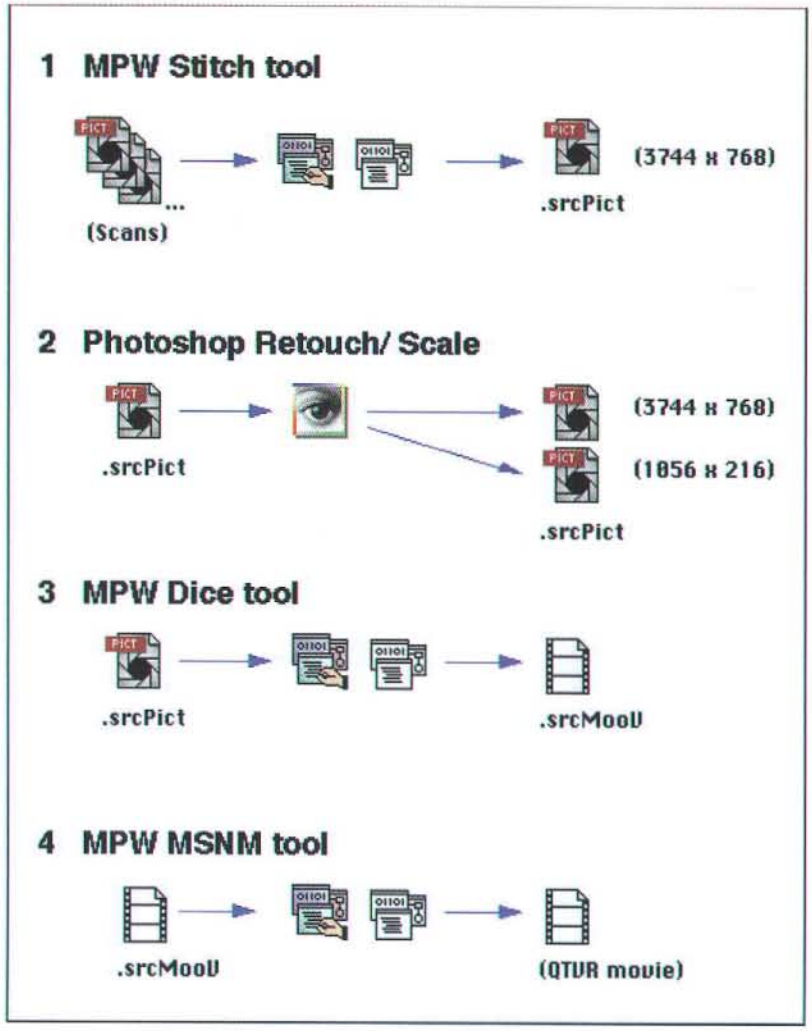


Figure 5. The QuickTime VR software process (24mm capture lens)

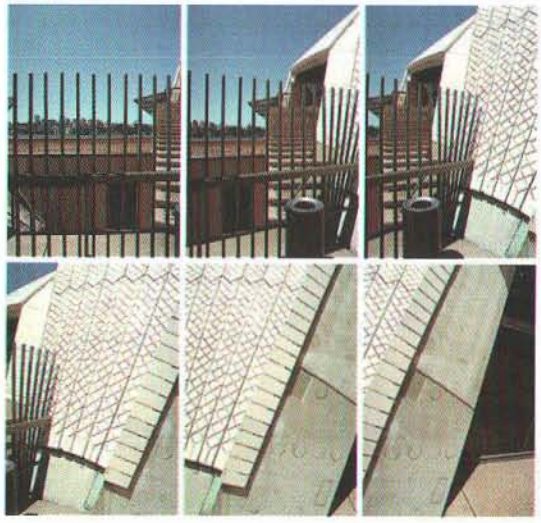


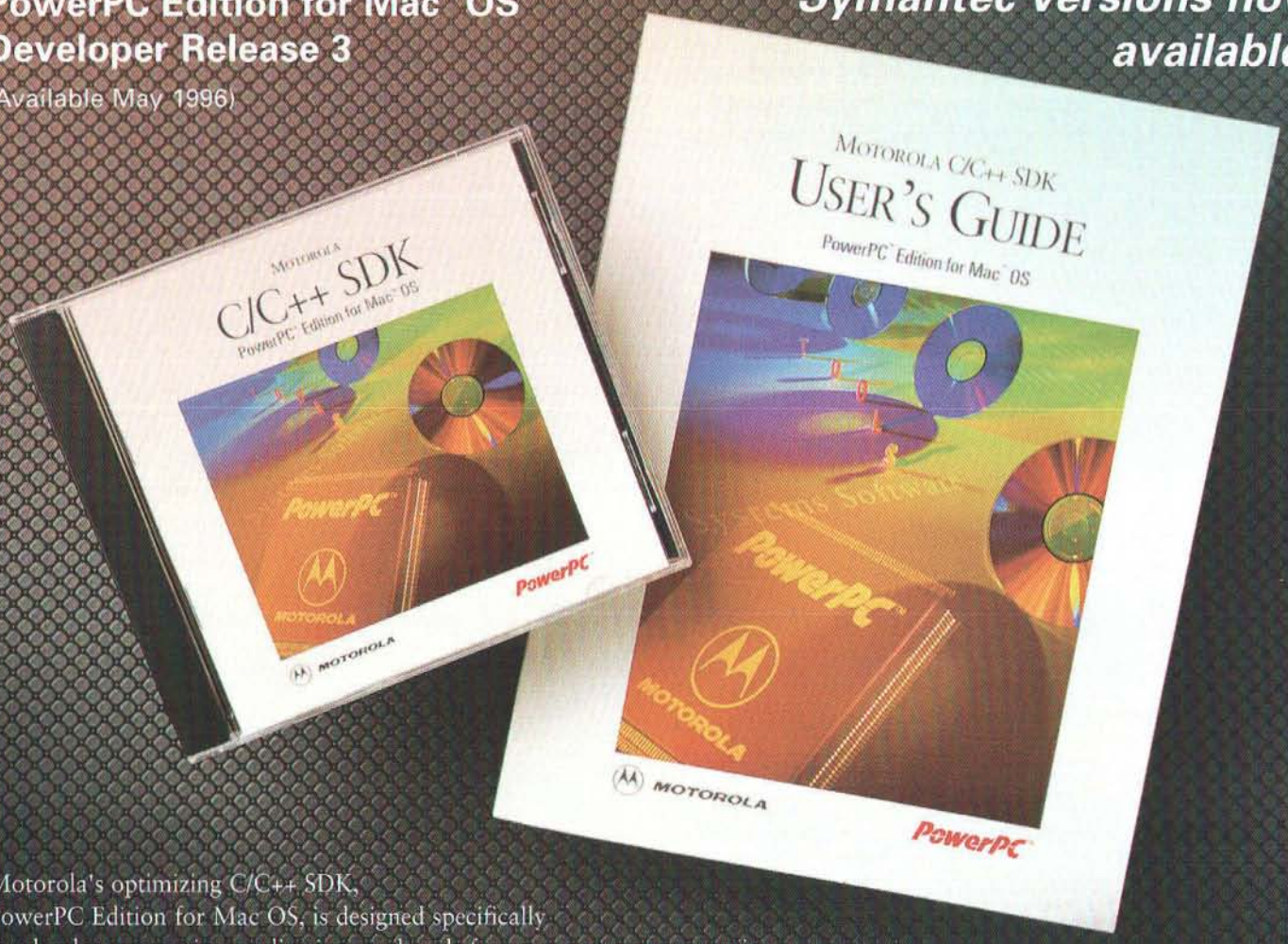
Figure 6. Six images prior to stitching

Unlock the power of PowerPC™ microprocessors.

Motorola C/C++ SDK

PowerPC Edition for Mac™ OS
Developer Release 3
(Available May 1996)

Codewarrior and Symantec versions now available!



Motorola's optimizing C/C++ SDK, PowerPC Edition for Mac OS, is designed specifically for developers creating applications and tools for computer systemsrunning Mac OS. The Motorola compiler works with Metrowerks CodeWarrior, Symantec C/C++ and Apple Macintosh Programmer's Workshop (MPW®) to offer a powerful set of development tools within a complete development environment.

The SDK features the Kuck & Associates Preprocessor (KAP™) — a powerful code restructuring tool designed to improve performance across a broad range of application programs. The Motorola C/C++ SDK delivers world-class code optimization for any specific PowerPC microprocessor, while ensuring compatibility across the entire PowerPC family of microprocessors.

Call 1-800-347-8384 or 1-512-891-2999.

Internet: ppcinfo@risc.sps.mot.com http://www.mot.com/PowerPC/

PowerPC™ MOTOROLA

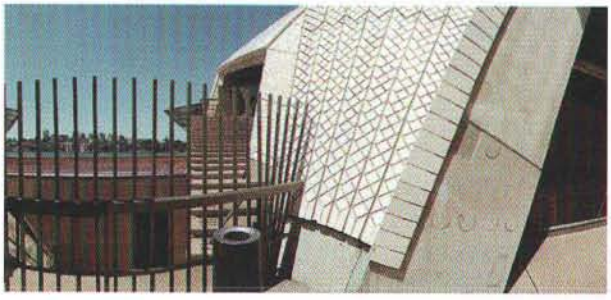


Figure 7. The six images after stitching

The advantage of doing it this way is that you can use almost any camera to photograph a scene. The disadvantage is that you have to employ a lot of computing grunt to make it happen. Hence the fast PowerPC.

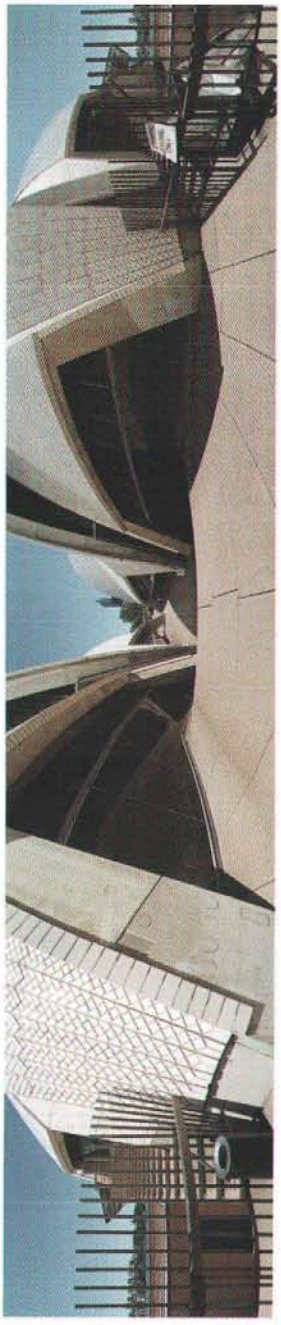
I use the following in my MPW worksheet to do a stitch. Although Apple supplies MPW script files to make the stitching process "easier", I prefer to call the stitch tool directly so I can see exactly what parameters are fed in.

```
# STITCH
#
# Because of 24mm lens, FOV = 60 & outHeight = 3744
#
set scansPath       "Schnieder HD:myQTVR:Raw Scans f"
set panOutFolder    "Schnieder HD:myQTVR:WIP"
set outName         "aznLatest.xx"
set scanFilenames   "01-16"

set myFOV       60
set myOutV      768
set myOutH      3744

stitch -fovy {myFOV} ∂
   -offset 250 0 -range 60 30 -rotate -90 ∂
   -dir "{scansPath}" -files "{scanFilenames}" ∂
   -wrap -cropOut -outHeight {myOutH} -outWidth {myOutV} ∂
   -sharpen -blend -fill -show ∂
   -out "{panOutFolder}:{outName}.srcPict" ∂
   #-i      #Uncomment for INTERACTIVE stitching
```

As Apple assumes a 15mm lens for all its MPW scripts and tools, adjustments have been made to accommodate the 24mm lens. Specifically, the Field Of View (`fovy`) value has been reduced from 97° to 60° and the offset and matching range have been widened to allow for the 24mm lens's smaller vertical and horizontal coverage.

The `outHeight` parameter has also been increased to reflect the greater number of shots required to cover a 360° view. The value of 3744 is the nearest multiple of 96 (4 times 24) to the output height produced whenever a stitch is done "raw" (i.e., without height/length concatenation). You need to pay attention to the height and width dimensions, as the dicing tool will fail if you get them wrong (see below). RTFM.

I recommend sticking with Apple's `outWidth` of 768 for the sake of RAM and movie run-time efficiency. Although you could work with larger image sizes, in practice the final movie quality does not appear to benefit from it. Likewise, the `sharpen`, `fill` and `cropOut` activities could be performed later in Photoshop, but again little practical benefit has been found in doing so.



← Figure 8. The final 360° stitched sideways panorama

You will notice that the script has provision for doing stitches interactively. This allows you – rather than the software – to determine precisely where images should be joined. Some developers always stitch with this option on; I prefer instead to stitch automatically at first and then re-stitch interactively only if there are problems.

After highlighting the script lines in MPW, you hit the Enter key and then watch as the stitcher does its magic and pieces together the separate scans into one panorama (the `show` switch above puts up a window during stitching to display the panorama being built). As mentioned earlier, it takes roughly four minutes for this to happen on a PowerPC 7500, at the end of which the resulting 8MB file (see Figure 8) is placed into the `panOutFolder`. Notice how the image is turned onto its side? This is deliberate and is required by the dicing tool.

Retouching the stitched image in Photoshop

As no scan is perfect, you will always have to adjust the image gamma, balance the colours, and remove dust spots. Similarly, no stitch is perfect, and sometimes you will also have to touch up any blend "failures" (see Figure 9).

← Figure 9. The white box encloses a stitch "failure"

Another thing you can (and should) do in Photoshop is produce scaled-down versions of the panorama for low-res versions of the QuickTime VR movie. An image size of 1056×216 will compact well into a movie small enough to be usable on the Internet (see below).

Dicing the Panorama

As an intermediate step, you must run the stitched panorama through the dicing tool to convert it from a PICT file into a QuickTime movie made up of 24 compressed frames.

The compression applied is of the Cinepak `cvid` kind. It is particularly aggressive (8MB becoming 800KB) and results in a moderate degree of image deterioration – so there is little point in being overly precious when retouching the panorama.

For the dicing step to work properly – as well to enable your movies to be playable across platforms – you must make certain that the height of the panorma is exactly divisible by 96 and its width exactly by 4. In addition, if the height is not exactly divisible by 24 then you will get a dicing error and you will have to re-stitch (or go back to Photoshop) to re-scale the image.

Because this is a straightforward step, I call the "SrcPictToMovie" MPW script supplied by Apple:

```
#DICING
#
set myBasename    "aznLatest.xx"
set myRoot        "Schnieder HD:myQTVR:WIP"

SrcPictToMovie "{myRoot}:{myBasename}.srcPict"
    "{myRoot}:{myBasename}.srcMooV"
```

On a PowerPC this step requires 20 MB of RAM and takes three minutes. On 68K machines it takes eight times longer (!).

Making the QuickTime VR movie

The final step is the conversion of the standard QuickTime movie produced by the dicing tool into the specially formatted QuickTime VR version:

```
#SINGLE-NODE MOVIE (24mm lens, WIDE size)
#
set myRoot      "Schnieder HD:myQTVR:WIP"
set myBasename  "aznLatest.xx"
set myPan       32
set myWind_H    400
set myWind_V    240

msnm "{myRoot}:{myBasename} (wide)" -source ∂
"{myRoot}:{myBasename}.srcMooV" ∂
  1 24 -vPanRange {myPan} -{myPan} ∂
  -windowSize {myWind_H} {myWind_V} ∂
  -defaultView 0 0 50
```

Again, because of the non-15mm lens, some of the parameters have been adjusted in order to get the perspective right. The `vPanRange` has been reduced from the more usual ±42.5 to ±32 degrees to allow for the shallower field of view for the 24mm lens. In addition, the default size of the completed movie has been enlarged slightly from the standard 320×200 to 400×240. The `0 0 50` parameters tell the tool where to set the default pan angle(s) and zoom ratio.

This step only takes a few seconds and results in an 800KB movie.

The Completed QuickTime VR movie

I lied earlier because there is still one more step before the movie is completed! You must open it in the QTVRPlayer application which comes with the tools and re-save it as "Self

contained" and "Playable on non-Apple computers". This final step will allow our QTW-equipped WINTEL friends to see what they are missing out on.

QuickTime VR On The Internet

At 500–800KB, a "hi-res" QuickTime VR movie is simply too large to put on the Internet. Although there are many ways of going about it, the following works well in producing scaled-down low-res versions (assuming a 24mm capture lens):

- In Photoshop, save a copy of the stitched panorama.
- Re-size it to be 1056×216 (note that 1056 is exactly divisible by 96 and 216 by 4).
- Run the dicing and "msnm" tools on this smaller image, setting the "msnm" default window size to 240×140.
- Re-save the movie as self-contained and playable on non-Apple computers, giving the resulting file either a `.mov` or `.qvr` extension.

The image quality isn't incredible, but at a 98KB size, the resulting movie is small enough for anyone to download.

On my home page I have also provided a small drag and drop application to allow users to restore the creator/file type of the downloaded movie to QuickTime VR's `'vrod'/'MooV'` – which makes sure users will be able to play the movies no matter what happens!

Other than to supply movies for others to view, the current state of the technology does not allow you to do much else. You cannot launch URLs from hotspots embedded in the movies, neither can you play sounds or link to graphic or text files – for this you need to construct MacroMedia Director, Apple Media Tool 2.0 or Hypercard extravaganzas.

QuickTime VR will aquire a more formidable net presence, though, once a "C" API is released. When that happens, Netscape plug-ins can be written and VRML will be `0xDEADBEEF`.

The Future of QuickTime VR

According to postings to the QuickTime Mailing List (see below), a phalanx of Apple DTS Engineers is working furiously to complete a "C" API similar to that for QuickTime. Currently (February, 1996), only Hypercard and Director XCMDs are supplied with the authoring tools – meaning that if you want to incorporate your movies with anything else, you can only do so via environments which support these externals (version 2.0 of the Apple Media Tool features built-in support).

When (if?) the "C" API becomes available, you will be able to integrate panoramas with anything you like. You could launch URLs, play sounds, link to movies or still pictures, or generally write your own player applications which do *exactly* what you want. Which is why we became developers in the first place, right?

To be fair, Apple has recently done the right thing in lowering the price of the Authoring Tools from a stratospheric

US$2000 to a more reasonable US$500. As of December, 1995, they have also waived most of the run-time license requirements. The MPW requirement should go and apparently will – with the next(ish) version of the authoring suite.

Meanwhile, we wait...

See Also

Apple's QuickTime VR site should be your first port of call for up-to-date information for developers, the latest versions of the QuickTime VR players and links to various organisations working in this area.
http://qtvr.QuickTime.apple.com

The QuickTime Development mailing list features extensive discussion of QuickTime VR related issues – in fact there is often so much that it swamps the more general QuickTime related postings.
listproc@solutions.apple.com
[No subject, with this message:]
subscribe QuickTime-dev (Your Organic Name)

Here is a good overview of the what and how of QuickTime VR. Includes a comparison with Microsoft's "Surround Video". Tom R. Halfhill, "See You Around", *Byte Magazine*, May 1995, 85–90.
http://www.byte.com

Manufacturers of QuickTime VR camera mounting accessories.
Kaidan@aol.com
http://www.kaidan.com

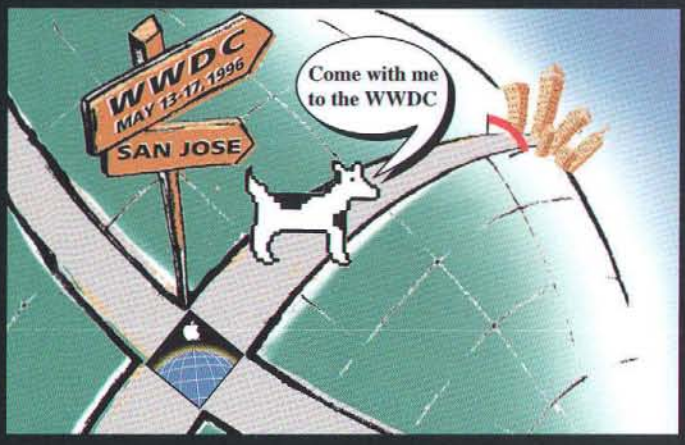
Omniview's "PhotoSphere" competition to QuickTime VR. Instead of a wide angle lens you use a super-expensive "fisheye" to capture image hemispheres.
http://www.usit.net/omniview

The Man From Warrimoo's home page featuring ten or so low-resolution QuickTime VR images of Sydney and environs.
http://www.zeta.org.au/~aznemeng

http://www.info.wwdc.carlson.com

Apple Worldwide Developers Conference

The Adventure Continues.



Feel the excitement. Be part of the adventure. Come to the 1996 Apple Worldwide Developers Conference (WWDC), where Apple's next-generation operating system signals a new Renaissance among Apple developers. You'll also meet the third-party development allies of tomorrow, and Cyberdog, the latest OpenDoc-based Internet solution tool suite. See what's in store for Newton and Pippin platforms, communications and collaboration capabilities, and multimedia technology.

Conference and registration information is available on the World Wide Web at: **http://www.info.wwdc.carlson.com** or by fax-on-demand service at **(800) 510-4529.**

Interested in sponsoring WWDC vendor exhibits? **Call (415) 283-3176**

1996 Apple Worldwide Developers Conference

May 13-17

San Jose Convention Center

Developer

Our competitors' habit of charging for each piece of their installer program is a bit puzzling.

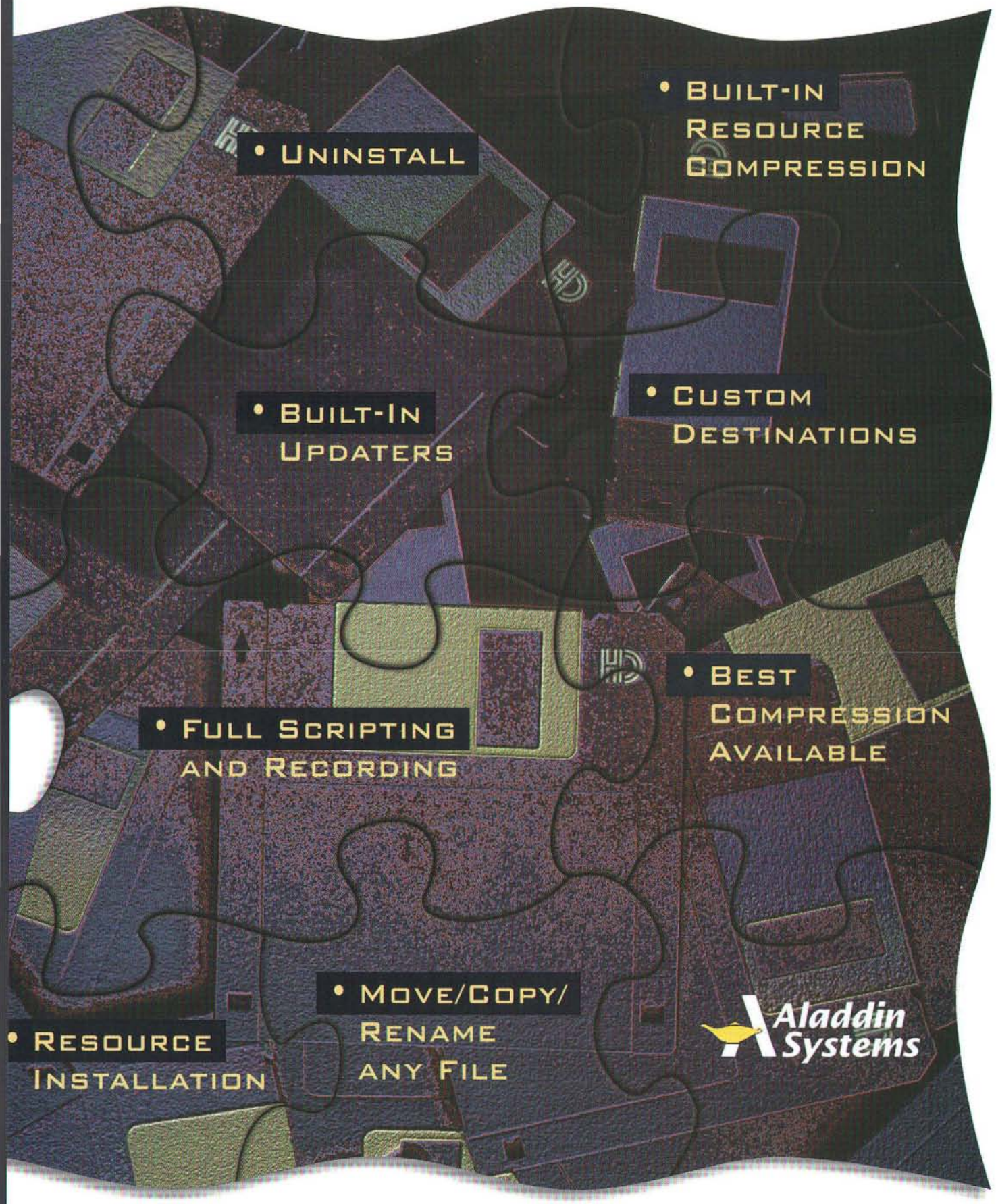
- CREATE UP TO 128 EXPERT-INSTALL "PACKAGES"
- IMPROVED DEVELOPER CUSTOMIZATION

The reason we call StuffIt InstallerMaker 3.0 the Complete Installation Solution is because we include **everything you need** to prepare a professional package of files for installation on your user's machines. An installer, updater, and an uninstaller are just three components included for the same price one of our fine competitors charges for just the installer. Using our installer reduces technical support calls due to end-user errors during installation, saves disks (if distributing on floppies) or download time (if distributing on-line). You will save more than the cost of the Installer license by significantly reducing distribution costs. And the puzzle will be solved.

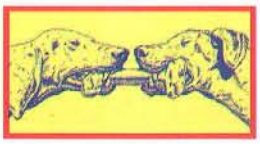
You'll have every piece in place.

Download a **free,** fully-functional copy of StuffIt InstallerMaker 3.0 from www.aladdinsys.com, or call (408) 761-6200 and ask for Developer Sales.

STUFFIT INSTALLERMAKER 3.0


• Uninstall
• Built-in Resource Compression
• Built-In Updaters
• Custom Destinations
• Full Scripting and Recording
• Best Compression Available
• Move/Copy/ Rename any File
• Resource Installation
Aladdin Systems

PROGRAMMER'S CHALLENGE



By Bob Boonstra, Westford, Massachusetts

EDGE DETECTOR

This month's Challenge is to write a small image-processing application that scans a color image and identifies the boundaries of possible objects in that image. Applications for such a program might include image enhancement, special effects, or pattern recognition, although those applications would use a more sophisticated approach for detecting edges than we will be implementing for this Challenge.

The prototype for the code you should write is:

```
typedef enum {
  redOnly=1, greenOnly, redAndGreen, blueOnly,
  redAndBlue, greenAndBlue, redGreenAndBlue
} EdgeType;

void EdgeDetect(
  PixMapHandle pMapH,           /* find edges in this PixMap */
  BitMap *bMap,                 /* store edges in this BitMap */
  unsigned short threshold,     /* color separations >= dist create an edge */
  EdgeType eType                /* which color components to look at */
);
```

Each pixel in the `PixMap` should be compared to the eight (or fewer) adjacent pixels differing in position by up to one row or column. If the pixel color is sufficiently different (as defined below) from any of the adjacent pixels, then the bit in the `BitMap` corresponding to that pixel should be set to 1. Otherwise, the `BitMap` bit should be set to 0. Obviously, pixels located in the first and last row and column will have fewer than eight adjacent pixels.

Whether two pixels differ by enough to constitute an edge is determined by comparing their rgb values. The distance between two pixels is the root-mean-square difference between the color components of their rgb values, considering only those components specified in the input `EdgeType`. For example, if the `EdgeType` is `greenOnly`, then the distance between two pixels is the absolute value of the difference in the green components of their colors. If the `EdgeType` is `redGreenAndBlue`, then the distance is the square root of the sum of the squares of the differences of the red components, the green components, and the blue components.

As a specific example, suppose we have two pixels with (red, green, blue) values of `(0x1000, 0x2000, 0x4000)` and `(0x2000, 0x5000, 0xB000)`. The distance between these two pixels is:

```
redOnly:          0x1000
redAndGreen:      0x3298=sqrt(0x01000000+0x09000000)
redGreenAndBlue:  0x7AE5=sqrt(0x01000000+0x09000000+0x31000000)
```

Two pixels define an edge if their distance is greater than or equal to the `threshold` parameter. The `threshold` parameter is deliberately declared to be an unsigned short, even though pixels can differ by a greater amount. Since the definition of distance is symmetric, the bits corresponding to both edge pixels would be set in the `BitMap`.

The `BitMap` will be allocated and initialized for you by the calling routine. The storage pointed to by the `BitMap baseAddr` will also be allocated and initialized to zero. The bounds rectangles will be the same for the `BitMap` and the `PixMap`. Your code needs to deal with pixelSize values of 8, 16, or 32, with each case being equally weighted in the scoring. For `PixMaps` with indexed pixels, you will obviously need to look at the color table to find the rgb value corresponding to a given index. In the 16-bit case, you should follow the rules for converting a 5-bit color component into an 8-bit RGBColor component value (i.e., replicating the 3 most significant bits and appending them to constitute the least significant bits of the 8-bit component).

This will be a native PowerPC Challenge, scored using the latest Metrowerks C compiler. (No C++ or Pascal this month.)

THE RULES

Here's how it works: Each month we present a new programming challenge. First, write some code that solves the challenge. Second, optimize your code (a lot). Then, submit your solution to MacTech magazine. We choose a winner based on code correctness, speed, size, and elegance (in that order of importance) as well as the submission date. In the event of multiple equally desirable solutions, we'll choose one winner (with honorable mention, but no prize, given to the runner up). The prize for each month's best solution is a $100 credit in the MacTech Mail Order Store and a limited-edition, "The Winner! MacTech Programmers Challenge" T-shirt (not available in stores anywhere).

Unless stated otherwise in the problem statement, the following rules apply: All solutions must be in ANSI compatible C. Use only pure C code. We disqualify entries with any assembly in them (except for challenges specifically stating otherwise). You may call any Macintosh Toolbox routine (e.g., it doesn't matter if you use NewPtr instead of malloc). We test entries with compiler options set to disable FPU use (for 680x0 code) and to enable all available speed optimizations. The compiler to be used and the target instruction set (680x0 or PowerPC) will be stated in the problem. **Limit your code to 60 characters per line;** this helps with e-mail gateways and page layout.

We publish the solution and winners for each month's Programmer's Challenge two months later. All submissions must be ***received by*** the 1st day of the month printed on the front cover of this issue.

You can get a head start on the Challenge by reading the Programmer's Challenge mailing list. It will be posted to the list on, or before the 12th of the preceeding month. To join, send an email to macjordomo@listmail.xplain.com with the message "sub challenge-A YourName".

Mark solutions "Attn: Programmer's Challenge Solution" and send it by e-mail to one of the Programmer's Challenge addresses in the "How to Communicate With Us" section on page 2 of this issue. Include the solution, all related files, and your contact info.

MacTech Magazine reserves the right to publish any solution entered in the Programmer's Challenge. Authors grant MacTech Magazine the exclusive right to publish entries without limitation upon submission of each entry. Authors retain copyrights for the code.

Entries Due Ten Days Earlier

Although two issues may seem like a long time to wait for the results of the Challenge, it has always been a challenge (no pun intended) to complete the scoring of results in time for publication two issues later. We have been searching for a way to allow a little more time for evaluating the entries and writing the column without introducing any additional delay between publication of the problem and publication of the solution. The Challenge mailing list has allowed us to deliver the Challenge to readers on a predictable schedule wherever they live, regardless of variations in mailing dates. We are going to use the mailing list to advance the due date for Challenge solutions, without reducing the amount of time available for solving the Challenge. Starting with this month's contest, Challenge entries will be **due earlier, on the 1st** of the month printed on the front cover. We will mail the problem to the mailing list on the 12th of the preceding month, also about ten days earlier than before.

If you are not already a member of the Challenge mailing list, you can join the ~300 subscribers from 25 countries already on the list by sending email to macjordomo@listmail.xplain.com with the line "`sub challenge-A YourName`" in the body.

Two Months Ago Winner

The response to the Words the Reverse Challenge was overwhelming. I don't know if it was due to allowing C++ and Pascal entries, or to the relative simplicity of the problem, but I received a record 45 entries to this Challenge. The Challenge was to write code that would reverse the order of words in a block of input text while preserving intervening white space and special characters, and adjusting capitalization of the reversed words to match that of corresponding input words. It is appropriate that the first Challenge admitting Pascal solutions was won by a well-known proponent of Pascal (see *MacTech Magazine* 12.4 [April 1996] 70, and p. 20 of this issue). Congratulations to **Peter Lewis** (Perth, Australia), author of Anarchie, NetPresenz, ObiWan, and other shareware products, for submitting the fastest entry to the Words The Reverse Challenge.

The test cases included a number of short, untimed strings designed to verify correctness. To my surprise, more than one-third of the entries failed these tests (or crashed outright). People had problems with strings that contained a single word, with strings that began with punctuation, with words of a single letter, and with the middle word in strings that contained an odd number of words. Remember, correctness is the first requirement for your solution.

For the timing tests, I ran a set of cases averaging around 40,000 words per case, totaling upwards of 500,000 words and 3 million characters for all cases. I eliminated from the input any "words" that started with a digit, because the problem statement was silent on how to deal with capitalization in that case. A number of people, including the winner, chose to treat words starting with digits as capitalization-neutral, so that the word being exchanged with the digit-word retained its original capitalization. This was a very reasonable approach (and I wish I had included it in the problem statement), but since the problem was silent, the fairest thing to do was to eliminate this condition from the test data.

The experiment allowing multiple languages and compilers went reasonably well. Most people, as requested, either provided a project/make file to link their solution with C code, or specifically indicated which compiler they wanted me to use. For those C entries that did not indicate a preference, I used the Metrowerks C compiler. A few people submitted solutions for environments that either did not generate native PowerPC code (e.g., non-SPM THINK C) or did not link with C code (e.g., THINK Pascal). For those entries, I mapped them to the closest possible environment (SPM C and Metrowerks Pascal, in these two cases).

Here are the times, compiler selection, code size, and data size for the correct solutions. Numbers in parentheses are the cumulative point total for all previous Challenges, not including this one.

Name	time	compiler	code	data
Peter N Lewis (10)	525	MW Pascal	896	42
Ludovic Nicolle (4)	602	MW C	1240	8
Kevin M. Cutts (50)	607	MW C	656	20
Gary Beith (20)	626	MW C	2088	8
Ernst Munter (132)	630	MW C	680	560
Robert Marsa	650	MW C	552	59
Eric Lengyel (40)	669	MW C	368	140
John Nevard (17)	670	MW C	564	20
Wolfgang Thaller (4)	681	MW C	616	20
Randy Boring	687	MW C	104	32024
Bill Karsh (80)	695	MW C	1128	8
Kirill Medvinsky	703	MW C++	540	12
Mark Bassam Salem	705	MW C++	592	32
Tom Saxton (10)	710	MW C	360	422
Karl Anderson	716	MW C	604	536
Lars Farm	762	MW C++	904	70
David McLeod	842	MW C	500	1315
Erik Sea	884	MW C	760	532
Robert Leslie/Geoff Hulten	938	MW C	436	268
Björn Davidsson (4)	1122	MW C++	1020	20
Tom Stone	1180	SPM C	752	16
Gustav Larsson (87)	1270	MW C	784	536
Ryan Gronlie	1294	MW C	444	20
Michael White	1397	MW C	1924	130
Rishi Khan	1447	MW C	960	8
Richard Fattic	1576	MW C	908	8
Stefan C. Sinclair	1668	SPM MrC	1248	40
David Newport	2420	SPM C	784	16
Ken Slezak (10)	2468	SPM C	808	16

To help understand why the single correct Pascal entry was faster than all of the C entries, I hand-translated the winning Pascal code into C, compiled it with several C compilers, and compared the results. I turned on all speed

optimizations in each case, and optimized for the 604 processor when the compiler supported that option. Since conventional wisdom is that C is more efficient than Pascal, I expected to find that the winning algorithm would be faster in C than it was in Pascal. In fact, the results for two of the C compilers were essentially the same as the Pascal results, and one was measurably worse (for reasons that I did not have time to investigate). Here are the results of my test:

Environment / Language	execution time	code size
Metrowerks / Pascal	525	896
Metrowerks / C	533	868
SPM / Symantec C	536	712
SPM / MrC	611	1992

Top 20 Contestants of All Time

Here are the Top Contestants for the Programmer's Challenges to date, including everyone who has accumulated more than 20 points. The numbers below include points awarded for this month's entrants.

Rank	Name	Points	Rank	Name	Points
1.	[Name deleted]	176	12.	Kasparian, Raffi	42
2.	Munter, Ernst	134	13.	Vineyard, Jeremy	42
3.	Gregg, Xan	92	14.	Lengyel, Eric	40
4.	Larsson, Gustav	87	15.	Darrah, Dave	31
5.	Karsh, Bill	80	16.	Brown, Jorg	30
6.	Stenger, Allen	65	17.	Lewis, Peter	30
7.	Cutts, Kevin	57	18.	Landry, Larry	29
8.	Riha, Stepan	51	19.	Beith, Gary	24
9.	Goebel, James	49	20.	Elwertowski, Tom	24
10.	Nepsund, Ronald	47	21.	Lee, Johnny	22
11.	Mallett, Jeff	44	22.	Noll, Robert	22

There are three ways to earn points: (1) scoring in the top 5 of any Challenge, (2) being the first person to find a bug in a published winning solution, or (3) being the first person to suggest a Challenge that I use. The points you can win are:

1st place .....20 points
2nd place.....10 points
3rd place .....7 points
4th place.....4 points
5th place.....2 points
finding bug.....2 points
suggesting Challenge...2 points

Peter's solution is relatively straightforward, and he hints in the preamble that he might have done better if he had spent more time on it. One tip that you might glean from Peter's code is the way he allocates dynamic memory. First, he deals with small problems with memory allocated on the stack. Second, he uses `NewHandle` rather than `NewPtr` to allocate dynamic memory. `NewHandle` is faster than `NewPtr`, because `NewPtr` may move relocatable blocks around before doing the allocation to avoid fragmenting the heap. Peter also locks the handle before using it, which is always safe, but is not necessary unless your code does something that moves memory. Whether locking is necessary in this case depends on whether you believe the documentation that says `BlockMove` doesn't move memory. Here is Peter's winning solution:

Challenge.p

Peter N Lewis, peter@stairways.com.au

```
unit Challenge;

interface

  uses
    Types;

  type
    CharsArray = packed array[0..0] of byte;
    CharsArrayPtr = ^CharsArray;

  procedure ReverseTheWords(
    text: CharsArrayPtr;
    numCharsIn: longint );

implementation

  uses
    Memory;

{
  This is not really optimal, I felt compelled to send in a Pascal solution since I was
  one of the people who complained about the language bias.  I didn't have time to
  do this challenge justice.

  Method:
  * Allocate a block of memory equal in size to numCharsIn (if numCharsIn < 2048,
    we short circuit this to use a block of memory on the stack).
  * Initialize a 0..255 array to determine whether a character is  an alphanum (I
    could just use the ANSI ctype.p file, but without a macro call, there is a pretty
    big hit).
  * reverse the words from the source to our new buffer.  We move in from both
    ends, copying non-alphanums, and then swapping words and fixing the case.
  * BlockMoveData the buffer back to the source buffer.
  * Release the memory if we allocated any.
}

  procedure ReverseTheWords(
      text: CharsArrayPtr;
      numCharsIn: longint );
    const
      stack_space_size = 2048;
    var
      space: packed array[0..stack_space_size] of byte;
      buffer: CharsArrayPtr;
      memory: Handle;
      leftin, leftout, rightin, rightout, leftedge,
        rightedge: longint;
      i: longint;
      leftchar, rightchar: integer;
      alphanum_set:array[0..255] of Boolean;
  begin
    { allocate memory if needed }
    if numCharsIn < stack_space_size then begin
      memory := nil;
      buffer := @space;
    end else begin
      memory := NewHandle( numCharsIn );
      if memory = nil then begin
        DebugStr( 'Memory allocation failed!' );
        exit( ReverseTheWords );
      end;
      HLock(memory);
      buffer := CharsArrayPtr( memory^ );
    end;
```

```
{ init - I wish I could do this at compile time - Turbo Pascal can }
for i := 0 to 255 do alphanum_set[i] := false;
for i := 48 to 57 do alphanum_set[i] := true; {0..9}
for i := 65 to 90 do alphanum_set[i] := true; {A..Z}
for i := 97 to 122 do alphanum_set[i] := true; {a..z}

{ reverse }
leftin := 0;
leftout := leftin;
rightin := numCharsIn - 1;
rightout := rightin;
while leftin <= rightin do begin
  while not alphanum_set[text^[leftin]] & (leftin <= rightin)
                                          do begin
    buffer^[leftout] := text^[leftin];
    Inc(leftout);
    Inc(leftin);
  end;
  while not alphanum_set[text^[rightin]] & (leftin < rightin)
                                          do begin
    buffer^[rightout] := text^[rightin];
    Dec(rightout);
    Dec(rightin);
  end;
  leftedge := leftin;
  rightedge := rightin;
  while alphanum_set[text^[leftin]] & (leftin <= rightin)
                                          do begin
    Inc(leftin);
  end;
  if leftin > rightin then begin { central word, just copy, ignore case }
    for i := leftedge to leftin - 1 do begin
      buffer^[leftout] := text^[i];
      Inc(leftout);
    end;
  end else begin
    while alphanum_set[text^[rightin]] do begin
                { there is a sentinel now, we dont need to check leftin < rightin }
      Dec(rightin);
    end;
    leftchar := text^[leftedge];
    rightchar := text^[rightin+1];
    if ( leftchar > 57 ) & ( rightchar > 57 ) then begin
                                          { both letters }
      if leftchar > 90 then begin
        if rightchar <= 90 then begin
          rightchar := rightchar + $20;
          leftchar := leftchar - $20;
        end;
      end else begin
        if rightchar > 90 then begin
          rightchar := rightchar - $20;
          leftchar := leftchar + $20;
        end;
      end;
    end;
    buffer^[leftout] := rightchar;
    Inc(leftout);
    for i := rightin+2 to rightedge do begin
      buffer^[leftout] := text^[i];
      Inc(leftout);
    end;
    for i := leftin-1 downto leftedge+1 do begin
      buffer^[rightout] := text^[i];
      Dec(rightout);
    end;
    buffer^[rightout] := leftchar;
    Dec(rightout);
  end;
end;

{ copy buffer }
BlockMoveData( buffer, text, numCharsIn );

{ free memory if required }
if memory <> nil then begin
  DisposeHandle( memory );
end;
end;

end.
```

Dilbert® *by Scott Adams*

Reprinted by permission of UFS, Inc.

BOOK REVIEW

By Mike Scanlin

The Need for Speed

Learn the nitty-gritty of PowerPC optimization

Optimizing PowerPC Code:
Programming the PowerPC Chip in Assembly Language
By Gary Kacmarcik
Addison-Wesley, 1995
ISBN 0-201-40839-2, 694 pages (softback). $39.95.

I'm disappointed. It's just no challenge any more. It took me years of careful trial, error, repeated error, and determined study, to perfect my 680x0 optimizing skills to the point where I really understood the chip from a software point of view. I was looking forward to the same kind of challenge on the PowerPC (scrounging for obscure magazine articles, surfing the net looking for example code, writing and timing code three different ways, disassembling all the programs with good performance to see how they did it, etc.). But now that I've read this book, all the hard theory has been taken care of, and the only thing remaining is to do a few PowerPC assembly language projects and put the theory to the test. Mr. Kacmarcik has cut short my search for knowledge by writing a book which makes plain everything about the PowerPC chip, including the subtle pipeline and cache interactions that a true optimizer wants to know.

This book is intended for programmers with some high-level experience and at least a little experience with assembly language. It does not explain what hexadecimal means, for example, but it does define concepts like "latency" and "throughput".

The first nine of the sixteen chapters review in precise detail the entire PowerPC instruction set and architecture. The purpose of these chapters is to broaden the audience for this book. Anyone with PowerPC experience could skim these 170 pages in an hour or so. For the rest, though, it is a reasonable starting point. Unfortunately, there are too few examples for the descriptions of the individual instructions to be meaningful. It's like someone handing you a book on how to write poetry where the first hundred pages are a dictionary explaining all the words you can use in your poems but not really giving you the context or any examples to appreciate them. It's hard to separate the really important stuff (like everyday instructions, registers and concepts) from the stuff that was just put in for the sake of completeness. An uninitiated person who tries to understand it all will probably become overwhelmed. I can accept that these chapters are meant to be an introduction and a bit of a reference (in addition to the complete references in the appendices), but it's a little too much, too soon, in my opinion.

The next seven chapters, and especially Appendix D, are the reason to buy this book. They contain the info that is hard to find elsewhere. The chapter titles will give you a good idea of what you'll find:

10. Memory and Caches
11. Pipelining
12. PowerPC 601 Instruction Timing
13. Programming Model [C calling conventions]
14. Introduction to Optimizing
15. Resource Scheduling
16. More Optimization Techniques

Appendix D. Optimization Summary

The cache discussion reviews how set-associative caches work. This is good info that you can apply to designing your own caches in higher-level languages like C. It is interesting to read that cache simulations have shown nearly identical cache hit rates for caches with random line-replacement algorithms and caches with least-recently-used line-replacement algorithms. There are tidbits of useful information sprinkled throughout this chapter, such as the sentence, "According to the PowerPC ISA, the programmer should assume that the processor has a split (instruction/data) cache, and that the processor will not automatically keep the instruction cache consistent with data written via the store instructions (that is, with the data cache)." Writers of self-modifying code, beware.

Even though the cache discussion is complete, it illustrates a problem that several of the chapters have: it's missing down-to-earth examples. For instance, it says the 601 has "a unified 32K, eight-way set associative cache", and explains what that means technically, but it doesn't go on to tell me how far apart two addresses need to be before they map to the same cache line. If I'm working on an image-filtering application, it is really

useful to know what sizes not to use for `rowBytes` (to avoid thrashing the data cache) if my algorithm visits all the pixels down a vertical column.

The instruction timing chapter was one of my favorites. Here's an example of the kind of precision you can expect:

> The Multiply Low Immediate (`mulli`) instruction always takes five cycles in IE. The length of time that the other multiply instructions spend in IE is dependent on the data contained in rB. If the upper 16 bits of rB are all sign bits, then the instruction spends five cycles in IE, otherwise it spends nine cycles. This means that the lesser (in magnitude) of the two arguments should be placed in rB because there is a potential savings of four cycles if -2^15 <= rB < (2^15 - 1).

All your favorite timing topics are handled here along with micro-examples to illustrate each stage of the pipeline for the entire sequence of instructions. Topics include: branch prediction (taken and not taken), cache hits and misses, pipeline synchronization, pipeline stalls, misaligned data accesses, and more. Here's another example of the kind of details you'll find. This is from the discussion of instruction fetching:

> This may seem like a strange thing to affect timing, but the address affects where the data will be stored in the cache, and the cache timing is different when the request is from the upper or lower part of a cache line. If your timings always assume that you'll receive four or eight instructions at a time, you may be surprised when the code is timed on a real system.... For a critical loop, it might be worthwhile to place a few nops before the loop so that it fits nicely into a cache line.

The programming model chapter was good. I especially liked the explanation of how leaf routines that don't need more than 220 bytes of stack space don't need to allocate a stack frame (because, by convention, interrupt routines know not to use the 220 bytes above the current stack pointer – known as the "Red Zone" in *Inside Macintosh*). This chapter also discusses why you should not use the Load and Store Multiple instructions.

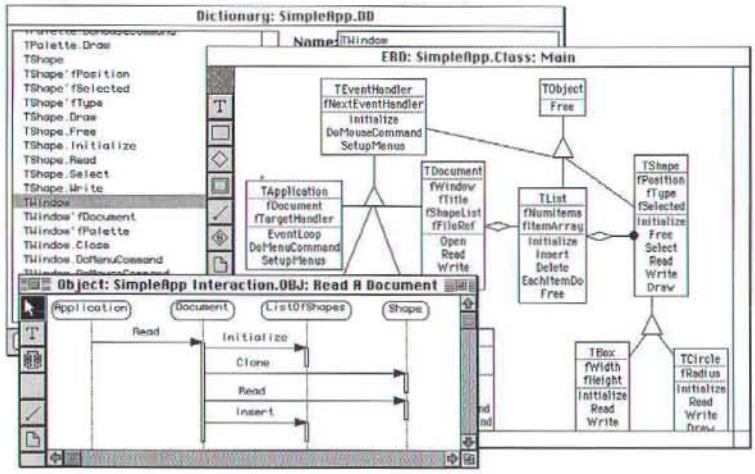
I must say I was disappointed that the chapter titled "Introduction To Optimizing" was only eight pages long. I was hoping that after plowing through 300 pages of details I would finally get to see 100 lines of before and after PowerPC assembly. But I didn't. So I kept plowing ahead and on page 317 I found out that, as a rule of thumb, I should always place two independent instructions between two branches that are taken (jumps to subroutines, perhaps). As I got further and further into the book I would find a gem like this every 20 to 50 pages. I couldn't help but think: "These are the really useful pieces of information; why can't he just list everything like this and give lots of examples?" Then I found Appendix D.

Appendix D begins on page 677 and ends on page 678. But those are the two best pages in the whole book. If you want to apply the 90–10 rule to reading this book and you only have time to read two pages, then you better make it these two – they are the "rules of thumb" to follow when writing PowerPC assembly code. If you do these things right then a large portion of your optimizing job will be done.

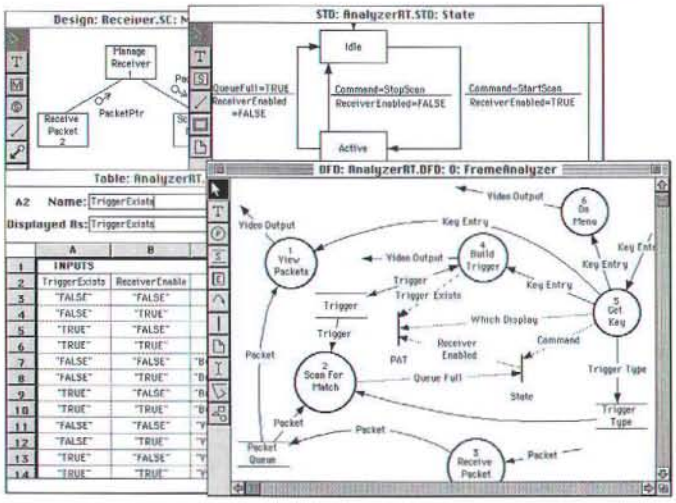
This is a great book. I was frustrated that I had to read almost 700 pages before I found the summary of tricks that I was looking for. But there are lots of little bits sprinkled throughout, such as the table on page 347 that shows how to multiply something by 3 through 10 with no more than 3 integer shifts, adds and subtracts. Mechanically, the book is beautiful to read. It is nicely typeset with fonts, font sizes and diagrams well chosen.

My biggest complaint is that I want to see real-world code examples (i.e. more than five instruction sequences) in action. I'd like the author to provide some high-resolution timer code so that I can time my own code and know if I've made a difference (how about a performance workbench to experiment with?). And I'd like to see things like a C program calling some performance bottleneck written in assembly so I could get a bigger picture of how all this code fits together in a real program. Nevertheless, if you have any interest in writing fast PowerPC code, you should buy this book.

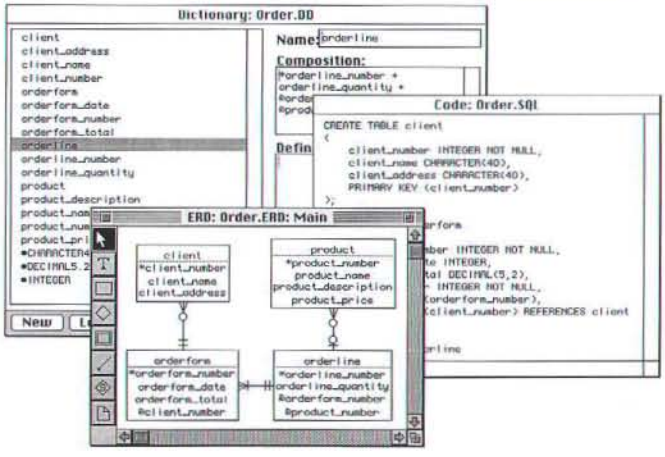
Draw and verify your design, generate code or reengineer existing code back to diagrams.



OOA/OOD includes OMT, Booch 94, Coad/Yourdon, Shlaer/Mellor...



Structured A&D using Yourdon/DeMarco, Gane/Sarson, Hatley/Pirbhai...



Data modeling and SQL generation for Information Engineering, Chen...

MacAnalyst & MacDesigner

Software Engineering

Structured Analysis & Design
Object-Oriented Analysis & Design
Real-Time & Multi-Task Design
Data & Screen Modeling
Integrated Code Editing & Browsing
Multi-User Dictionary & Requirements
Code to Design for C++, Pascal, Fortran...
Design to Code for C++, Pascal, Fortran...

New in Version 5.0

Jacobson, Fusion, Use Cases, Harel Statecharts, Interaction Diagrams and more

Products are available by single, site or educational license and supported with on-site training, update service, free newsletter and technical phone support. Products include sample documents, tutorials and online help.

Product Options

MacAnalyst	$ 995
MacAnalyst/Expert	$1595
MacDesigner	$ 995
MacDesigner/Expert	$1595
MacA&D	$2995
Translator	$ 495

System Requirements

Macintosh System 7 with 8 meg RAM or UNIX with Apple's MAE software. Winner of the 1994 CIO Magazine Readers' Choice Award, MacAnalyst and MacDesigner are used by thousands of developers worldwide for personal computer, mainframe and embedded software projects.

Call now for free technical brochures!

Excel Software

515-752-5359

P.O. Box 1414 • Marshalltown, IA 50158
casetools@aol.com • Fax: 515-752-2435

By Tantek Çelik and David Curbow

Rethinking the Interface

Getting the look and feel of a container application

Many OpenDoc parts are being created, and much has been written about how to do so; but very little has appeared about how to "host" OpenDoc parts – that is, how to be a container application. Indeed, as of this writing, only five container applications are even known to exist. Issues both of Human Interface and of programming technique are at last being worked out; and here, two leading authorities in the field expound for the first time what an existing application needs to do in order to become a container application.

Adding OpenDoc support to your application requires rethinking your application's human interface design a bit. In many ways, adding OpenDoc embedding is no more intrusive than adding QuickTime movie embedding. However, because of OpenDoc's generality, it relies on sharing various application-owned structures. In short, these are: document files, menus, windows and events, clipboard, drag and drop, and, of course, the application heap. Most of these affect the human interface of your application.

OpenDoc part editors assume there is a document shell which provides certain functions, such as default menus and a document model. Parts also assume that either they are the root part of a document or they are embedded into a containing part. In the latter case, they will interact with this containing part in certain ways, for example through frame negotiation. It is also important that the containing part provide access to the Part Info dialog for all the parts it contains. We did our best to strike a good balance between ease of development for the part editor, on the one hand, and minimal change for the container application, on the other. This article both delineates the practical minimum of what an application *must* do to adapt its human interface to support OpenDoc embedding, and explores what an application *could* do to be as seamlessly integrated as possible into the OpenDoc user experience.

Document Files

Your application can keep its current document format, as long as it has the ability to store an object of arbitrary length at some application-determined offset at save time. Alternatively, your application can make use of OpenDoc's structured storage model, and encapsulate your current document format inside an OpenDoc stream. This latter approach makes your documents more forward compatible – you will be able to write part editors to read and write the exact same format. The *MacTech Magazine* 12.1 (January 1996) article titled "OpenDoc: Contain

Tantek Çelik is the *MacTech Magazine* OpenDoc Contributing Editor. He is also a founder of 6prime corporation, a software consulting firm dedicated to pervasive OpenDoc and MacOS adoption among developers and users. In a previous life he was one of the two technical leads for the OpenDoc project at Apple Computer. He worked on the design and implementation of OpenDoc and its related technologies (CALib and PartMaker) from the beginning through the completion of OpenDoc 1.0 and its GM release on DR4.

David Curbow has been the Human Interface Lead on the OpenDoc project at Apple Computer since its beginning. Dave makes sure that the eager OpenDoc engineers don't get too out of line with their implementation of the portions of OpenDoc which affect the human interface. He also edits the OpenDoc Human Interface FAQ.

Yourself", by Eric Soldan, does a great job of illustrating how container application support affects your document storage model at the implementation level; no need to repeat here.

If you make changes to your file format, you may be wondering if you should also modify your suite of icons, perhaps to convey the idea that your application supports OpenDoc. We recommend against that. You probably didn't change your icons when you added support for QuickTime, so you shouldn't make any changes now.

Menus

There are a few differences between your application's menus and those of OpenDoc. There are very few changes you'll need to make and we'll discuss them in detail. Here are OpenDoc's default menus:

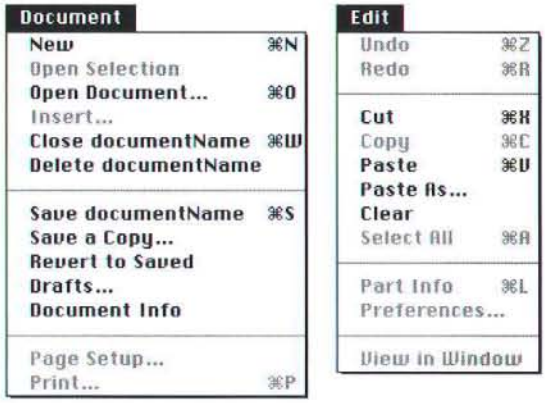


Figure 1. OpenDoc default menus

The Document Menu

The first difference people notice between OpenDoc and today's applications is that OpenDoc's default first menu is called **Document** instead of **File**. We believe that "File" is overused today. It often refers to documents, control panels, network browsers, utilities that have nothing to do with files, etc. In OpenDoc, we use a "Document" menu name for documents, and we will use other menu names in the future.

For now, we recommend that you continue to use **File** as your first menu. That way, **Document** is a clue to users that they are working with an OpenDoc document being handled by a part editor, as opposed to an application.

The second difference people notice is that there isn't a **Quit** command in the first menu. We got rid of the **Quit** command in OpenDoc for a very important reason – orphan menubars. When the user closes the last document in most applications today, the menubar remains behind, and this often causes users to be confused. We chose to fix this problem by using a single process per OpenDoc document and closing that process when the user closes the document. Of course, to use this solution, the time needed to open each document must be considerably smaller than what it takes to launch a full-blown application. (Yes, we know opening a document isn't fast

Get Your Name "In Lights"

Here at *MacTech Magazine*, we rely heavily on outside writers for most of the material that appears in our pages. If readers did not participate in the magazine, sending us their ideas and taking the time to write articles, there would be no *MacTech*. We like to think of *MacTech* as an ongoing dialogue amongst members of the Mac programming world: we facilitate the discussion, but it's the readers who carry it on, by responding to what they read and to their own programming experiences and interests, in writing. Sometimes we know that we need something specific covered, and we approach someone to write an article on that subject; and we do write a few columns in-house each month. But it is reader contributions, in the form of letters, tips, and especially articles, that give the magazine its relevance, its character, and its appeal.

So *MacTech Magazine* is not a staff of writers sending a constant stream of one-way messages outwards; it's a living, evolving network of readers conversing with one another, educating one another, sharing their knowledge, their experience, their interest, their trials and tribulations and joys and successes in the constantly unfolding story of programming the Macintosh. *MacTech Magazine* doesn't just happen: it's what the community makes it. If we carry reports of future trends and technologies, if we teach useful methods, if we review new books and tools, if we provoke thought, provide help, ride the wave of current interests and concerns, it is only because we reflect the thoughts of our readers, who speak through our pages.

You are invited to involve yourself in this exciting conversation amongst readers. You may be working at the cutting edge of programming technology, as part of a heavily funded professional developer effort; you may be a lone hobbyist wrestling to create shareware for the sheer love of it. You may have been programming the Mac since its inception; you may have just switched over from Windows or Unix. You may work in C or C++ or Pascal or AppleScript, from scratch or in a framework. You may write big apps, small apps, custom solutions, extensions, code segments, for profit, for fun, for education, to solve one problem once. No matter who you are, no matter what your credentials may be, if you have a tale to tell, a trick to share, a technique to teach, we want you to consider joining the family of those who write for *MacTech*.

Don't just wait for a topic to be covered in *MacTech!* Don't just wish some technique would be explained better! Take responsibility! Write us an article yourself!

To write for *MacTech*, just send for our Writer's Kit. It's a Microsoft Word file containing the Styles you need to use, and giving lots of helpful advice and information, including all the legal stuff. You can let us know what you're writing about, if you want suggestions or feedback; this is helpful to us, because it lets us plan for the future. Or if you want to, you can just write the article and spring it on us when it's done. If we publish your article, you'll be paid for it!

Write to me, Matt Neuburg, at managing_ed@mactech.com (or one of the other editorial addresses listed on page 2 of the magazine). Ask me for a Writer's Kit! Send me an article! Get published! Make money! See your name in print up there alongside the famous denizens of *MacTech!* And, most important, take the future of *MacTech Magazine* into your own hands!

MacTech MAGAZINE

enough yet in version 1.0, and that's a focus for our upcoming releases.) Most of today's applications cannot use this solution, so we recommend keeping the **Quit** command.

Notice also that OpenDoc uses **Save A Copy...** instead of **Save As...**. Whatever you are using now is fine; don't change it. At this time, Drafts are not supported for Container Applications. If your application already has a sense of providing access to the Document's name, create/mod dates, comments and info, it is appropriate for your application to provide a **Document Info** menu item. Otherwise, omit this command from your **File** menu.

OpenDoc's **Close** menu item by default applies to the frontmost content window, which is not always the window containing the document. You may need to change the behavior of your application to match.

You may not be aware of the following "power user feature". If the user holds down the Option key and chooses **Close**, or types ⌘-option-W, your application should close all the windows of the document to which the frontmost content window belongs. We recommend adding this behavior, but it is not required.

The Edit Menu

The **Edit** menu also has a few changes from today. Today the Undo and Redo functions are usually handled by one menu command – **Undo**. OpenDoc supports multiple levels of Undo

and Redo, so it is necessary to have two commands. It is likely that your application only has the **Undo** command, but when an embedded part is active it will have **Undo** and **Redo** commands. This means that there can be some confusion, unless you change your application to also have both. Be aware that OpenDoc reserves ⌘R for **Redo** and we recommend that your application not use this command-key equivalent for any other operation.

Your application can support Undo in a couple of ways, but to get the best user experience, it should support multiple levels of undo. It is perfectly acceptable, however, to support only one level.

Given that your container application probably does support only one level of undo, there are three points we need to cover. First, it is important that your application not purge the undo stack every time the container is activated. Instead, purge the stack after the user does some undoable action in your contents. You should use your existing undo mechanism.

Second, you need to enable either the **Undo** or **Redo** commands, but not both. When the user invokes **Undo**, your application should disable **Undo** and enable **Redo**, and if the user then invokes **Redo** you should get back to the state things were in before **Undo** was invoked. This is essentially the same as what many apps do today, except that here, **Undo** and **Redo** are different menu items.

Third, you need to clear the **Redo** command whenever a new undoable action is to be recorded. Suppose that an action has been undone and the user then selects some content and invokes **Cut**. At this time, your application should purge the redo action, disable **Redo**, record the new undo action, and enable **Undo**.

On the other hand, if your application supports multiple levels of undo, you must use the OpenDoc undo stack in conjunction with the above. When the user does some undoable action in your container, put that action on the OpenDoc undo stack (as described in *OPG*). If the user activates an embedded part and then reactivates your container, your application needs to decide whether to enable the **Undo** and/or **Redo** commands; to do this, you look at the status of the OpenDoc undo stack (check out the `PeekUndoHistory` method of the `ODUndo` class).

Now we'll talk about the **Paste As...** command. This command allows the user control over whether pasted content is to be embedded as a new part, or merged into the currently active part. It also allows control over which *kinds* to paste and which editor to associate with new parts. This command can also be used to create links, which are based on Publish & Subscribe. We recommend that you leave this command out of your **Edit** menu.

The **Info** command applies to the current selection, and provides information about the selection. To make it more obvious what kind of info will be given to the user, the menu command changes its name according to the selection. For example, when a part is selected, the menu name is **Part**

Info, and when a link is selected, the menu name is **Link Info**. The intent is that users can find info about *any* content via an **Info** command. To make it easier for users to find this functionality, we recommend that editors and container applications reuse this menu command whenever possible. For example, when a sprite within an animation is selected, the menu command might be **Sprite Info**.

Not all applications provide this kind of information, so if no info can be given about the selected content, the menu command would be disabled. When the user selects an embedded part, your container application should change the menu name to **Part Info**.

If your application already supports an **Info** command, simply change that menu item to **Part Info** when a part is selected; there is no need to add a new menu item. Also, if your application already has a command-key equivalent for this **Info** command, don't change it. If not, and you have not already made use of **⌘L**, then go ahead and make it the command-key equivalent for your **Info** command. If you don't currently use **⌘L** in your application, please avoid adding it in the future – keep it reserved for the **Info** feature.

Most applications don't allow new views of their content to be created. Therefore, we recommend that you omit the **View In Window** command from your **Edit** menu.

Previously, we've recommended that your application have a **Preferences...** menu command in the **Edit** menu. OpenDoc follows that earlier guideline, but we recommend that you add the name of your application – just to make it clear which set of preferences will be presented. For example, the command might be **SurfWriter Preferences...**.

In your application, you may have menu items that are not in the OpenDoc **Edit** menu. You may be able to easily rearrange your menu items so that you follow the guidelines on p. 543 of the *OpenDoc Programmer's Guide* (hereinafter, *OPG*). Obviously you need to consider the impact on your documentation.

Your Application's Menus

Your application's current menus are probably fine as is. However, there are a couple of things to watch out for. We've already mentioned the Undo/Redo split and the two newly reserved command-keys, **⌘R** and **⌘L**.

In addition, command-key equivalents that you've assigned to operations in the **File** or **Edit** menus will be usurped by an embedded part. For example, suppose your application uses **⌘I** for some command, such as Get Info. Now suppose that an embedded text part uses **⌘I** for "Italics". When this part is active and the user types **⌘I**, the **Italics** command will be executed instead of **Get Info**.

Menus for Embedded Parts

Now that we've discussed the new commands that OpenDoc adds to the menubar, it's time to talk about implementing a menubar for embedded parts. As stated above, embedded parts expect to reuse the document shell's menubar

as a starting point. Therefore your application must fulfill this function. It needs to provide a "base menubar" which the part editors running inside your documents will use as a starting point for their menus. This consists of the "standard" Macintosh **File** and **Edit** menus, with a few more menu items as specified by OpenDoc Human Interface specification.

Part editors expect access to at least the following set of menus.

Standard Macintosh:

File: **Close**, **Page Setup...**, **Print**
Edit: **Cut**, **Copy**, **Paste**, **Clear**, **Select All**

OpenDoc additions:

File: **Open Selection**, **Insert...**
Edit: **Undo** and **Redo** as separate items, **Paste As...**, **Part Info**, **View in Window**, **Editor Preferences**

Once you have provided the base menubar for parts to use, you may add all your application- and content-specific menu items for your application's menubar. You may also remove menu items which your application does not support. Some of the menu items introduced by OpenDoc your application will need to keep, however.

Embedding New Parts

Many of today's applications already support some sort of embedding, even if it is only embedding of PICTs or QuickTime movies. In our examples, we'll concentrate on these kinds of applications, since they already have some sort of model for embedding objects in their content. Most such applications support embedding via the **Copy** and **Paste** commands. Some applications have tool palettes for creating new embedded objects (such as drawings). A few applications have recently added support for drag and drop. We recommend reusing these same mechanisms to add embedded OpenDoc parts. This was discussed in some detail in the May 1995 *Apple Directions* article, "How to Add Content to OpenDoc". You should also see p. 593 of the *OPG*.

Your application must support at least one and potentially all three standard mechanisms for embedding content (insert, copy/paste, drag and drop). The first is the easiest, and thus is the best place to start implementing OpenDoc embedding. Many applications have an **Import...** or similar command. In OpenDoc we decided that **Insert...** was easier to understand. However, your application may continue to use any existing **Import** command.

If you are considering converting your application to a container application, you may be worried that you must now support drag and drop. This isn't true. Users like drag and drop because it provides an immediate sense of control, and supporting it may make your application stand out, but there is no requirement to support drag and drop just because you want

your application to be a container application. We'll talk about implementing drag and drop a bit later in this article.

Windows

The most visible aspect of embedding OpenDoc parts is that your application gives the embedded parts some real estate within its document windows. To do this, you have to register each of your document windows with OpenDoc.

Events

When the user clicks in your application's content, your content (and application for that matter) are said to become "active". Your application needs to request the user input focus and install its menubar as the current one.

Parts do the same thing in response to clicks in their content. Menu and keyboard events automatically route to whatever part is active (has the user interface focus). If the part does not handle a menu event, the document shell gets a chance to handle it. Your application is the document shell from an embedded part's perspective, so you will still need to handle some menu events when embedded parts are active.

Selecting Parts

Users shouldn't be able to tell where the part boundaries are until absolutely necessary. Thus, selecting one or more embedded parts should be as easy as selecting other content within your application. However, it is more important to be able to easily activate a part than select it. So, a click on a part will always activate it instead of select it. All other selection mechanisms, such as selection by dragging, should apply to parts as well as other content. Selection of parts is covered in detail pp. 562–568 of *OPG*.

Clipboard

Typically applications have a local scrap which they use exclusively until a suspend or resume event occurs, at which time they synch with the system scrap. Because of OpenDoc, you must synch with the system scrap more frequently, so that you may properly exchange clipboard data with embedded parts. Whenever your application loses the user interface focus, you must export to the system scrap. Whenever your application acquires the user interface focus, either via the user clicking in your content area or by an OpenDoc part relinquishing the user interface focus (notification here also), you must import from the system scrap, or at least note that your local scrap is old and update it later when needed.

Drag & Drop

If you application does not support drag and drop, users may be a little surprised to see that dragging and dropping into already embedded parts works, but that in your application's content it doesn't. Supporting drag and drop is optional, but your users will thank you if you do.

If you implement drag and drop, you need to support files being dragged in from the Finder as well as dragging content to the Finder. When content is dragged to the desktop, you should create a new document on the desktop which is the same data format as the current document. This way, the user deals with only one document format, instead of having to also deal with clippings. OpenDoc introduces this simplification to the drag and drop model.

The Application Heap

OpenDoc's shared libraries plus the structures for one simple open OpenDoc document occupy about 100K of application heap. OpenDoc's shared libraries get loaded into temp mem, not the app heap, so the code size (i.e. PowerPC vs. 68K) does not affect this 100K. It is recommended that you raise your application's recommended minimum heap size by 100K accordingly. You may keep the current required minimum heap size for your application, as long as it is capable of disabling features (such as OpenDoc embedding) when low memory conditions result. If you want all features to be always available, then you should also raise the required minimum heap size for your application by 100K.

Conclusion

From this discussion it is pretty clear that OpenDoc support involves touching quite a few places in your code. However, it should also be noted that most of these changes are as simple as one or two additional lines of code. There certainly is some freedom with respect to how much you want to change your application's human interface to support OpenDoc embedding. Hopefully this article has conveyed an idea of what the range of freedom is, and has provided some help with what each of the human interface changes means for your application in particular. So go out there and figure out how you want your container application to look and feel.

6prime Corporation
http://www.6prime.com
OpenDoc
http://www.opendoc.apple.com
CALib and PartMaker
http://www.opendoc.apple.com/dload/developer.html
OpenDoc Human Interface FAQ
http://dev.info.apple.com/appledirections/mar96/opendocfaqs.html
OpenDoc QT demos
http://www.opendoc.apple.com/demos/HIdemos.html
MacOS
http://www.macos.apple.com
Further information on Container App development
http://www.6prime.com/containerapp.html
Apple Directions article
http://dev.info.apple.com/appledirections/may95/techopendoc.html

By Dave Mark

Discover Java

This month, the big news from Metrowerks is the release of their long-awaited Java development environment, timed to coincide with the start of WWDC. Metrowerks' version of Java comes in two flavors. If you already subscribe to CodeWarrior Gold, have no fear, the complete set of Java tools is included with the just released CW9. The complete Java toolset will also be packaged without all the C, C++, and Pascal tools under the name *Discover Programming with Java.*

I had a chance to speak with some of the engineers behind Metrowerks' Java effort. Without further ado, here's my conversation with Marcus Jager, Clint Popetz, Peter N Lewis, Tim Freehill, and Mike Lockwood...

Dave: Let's start off by talking about CodeWarrior. Can you explain the process of building, compiling, and running a Java project?

Marcus: Our primary goal with supporting Java on the Macintosh was to make the experience as Mac-like as possible. This means that we had to remove from Java the over-dependence on file system hierarchy and environment variables. The user shouldn't have to do anything more than add a pile of Java source files to a project and hit ⌘R.

Peter: We tried to make working with Java almost exactly the same as any other CodeWarrior target. So you make a new project (using a Java stationery document, or just select Java as the target). Then drag in your Java source files, and any zip files you need (you can weak link these if you know they will be available). Like the other targets, you use the Java project panel and select the type of output and other options. Then you can debug your program using the same source level debugger that CodeWarriors are used to.

Tim: Java has really fit smoothly into the IDE. The language-specific features were altered to support Java, so now you have Java keyword highlighting, and Java methods are parsed for the function popup. But using CodeWarrior to program Java really is not much different from using CodeWarrior to program C++.

Dave: What is the difference between an applet and an application? What does a Java application look like on the Mac?

Marcus: This is where something like Java causes terminal confusion in the naming of things. For the record: An applet is just a sub-class of the `applet` Java class, nothing special. Applet objects are what Web browsers use to embed Java code in web pages; this is what most people mean when they talk about Java. A Java application is a complete self-contained program that runs independently from the network and Web browsers. Currently these are executed in the standalone Java interpreter application, but there are better ways of doing this.

Marcus Jager is a YAWAMP (Yet Another Western Australian Macintosh Programmer). He started the great brain drain by defecting to the US a year ago to the auspiciously named "Mammoth Micro Productions". After that company went belly-up he has moved on to bigger things at Metrowerks.

Clint Popetz is someone who would apply object-oriented techniques to pacifying his daughter, if his wife would allow it. He believes that anyone who thinks C++ is ugly should try perturbation theory.

Peter N Lewis is a prolific author of fine Macintosh shareware and freeware products, including Anarchie and NetPresenz which are distributed by his company, Stairways Software Pty Ltd. He specializes in TCP/IP products but has been known to diversify into other areas. Peter is still trying to figure out how he got conned into helping out Metrowerks with the Java compiler – he suspects it may just be that he's having a lot of fun.

Tim Freehill has spent the last few years developing database tools, and has moved up to the challenge of creating top-notch Java tools at Metrowerks.

Mike Lockwood is the author of SourceBug and the cult classic VoodooMonkey debugger. At Apple, he worked on the Finder and the Apple Dylan project, and is now a covert debugger operations specialist at Metrowerks.

Clint: An applet can only be viewed within an "applet context". This context can be provided by a browser like Netscape, or a simple program like the AppletViewer. Applets are intended to be embedded in HTML pages. Since they are meant for dispersion via the internet, they are placed under pretty strong security restrictions with respect to disk access, loading native code, or accessing the network.

Peter: Using CodeWarrior you can build several other kinds of outputs. My favorite is a Macintosh droplet – this is just a tiny 68K application that asks the Java interpreter to execute the zip file stored in its data fork. If your target is a droplet then you can just choose **Run** from the **Project** menu, the project will be brought up to date and your code will be executed in the interpreter. If you add a `BNDL` resource you can then drag and drop files onto your droplet. Also, you may be able to include some native shared libraries and then use native classes to do processor-intensive or hardware-specific Mac-only solutions. So, for example, you might write some C code to interface to a scanner, package it up as a Java class and a shared library, and then do all the rest of the code using Java, perhaps using AWT as the interface, or writing some more native Mac code. I think this is going to be a lot of fun.

Dave: The standard user interface in the Java universe is defined by AWT (the advanced windowing toolkit). The AWT interface is definitely different than the standard Mac interface. For example, under AWT, each window has its own menubar. How is AWT implemented on the Mac?

Clint: Well, I mapped the AWT components onto PowerPlant `LPane` subclasses. So AWT buttons look like LStdButtons, etc. Since each AWT Frame (window) can have its own set of menus, I have each window put its menus in the bar when activated, and pull them out when deactivated. A bigger problem is that there is no equivalent of the Mac Human Interface Guidelines for the AWT; you can make your "OK" button say "Yessir," put it in the top left corner of the window, make it mauve, and make its font italic. And since many people writing Java code will not be used to the Mac, you can expect a lot of weird-looking Mac windows.

Dave: Do you think we'll ever see a mainstream Macintosh application written in Java? Perhaps based on a more Mac-like AWT with its own version of Constructor?

Marcus: Time for some marketing speak: "Metrowerks considers these to be important future directions." Java has opened up a world of possibilities, and it will be a while before it's clear what its true strengths will be. I think that Java is much more than the Internet and the Web. I would love to see mainstream Macintosh applications written in Java. I think the very strong type safety and automatic garbage collection would be a big step forward in programming practice and lead to better quality programs.

Dave: Can you call C code from inside a Java applet? If so, what is the binding mechanism that makes this possible? What security implications does this have?

Clint: Applets can use native C code, but only if it is has already been loaded by the Java virtual machine. So the virtual machine decides what native code is safe (like the native code that implements the AWT), and the applets can use this.

Marcus: The Java virtual machine calls C code from Java by linking to a shared library and calling the C functions contained in it. Since the virtual machine has no way of verifying what the shared library does, it relies on the user to install only libraries that they know are safe, and provides no automatic system for downloading them.

Dave: What impact will mixing Java and C/C++ have on my ability to debug my programs?

Mike: The CodeWarrior debugger will support debugging both C/C++ code and Java code simultaneously. The CW8 debugger can already debug 68K and PowerPC code simultaneously, and in CW9 we are adding Java support as a third target. You will be able to single-step through both C++ and Java code, display C++ and Java objects, and see both Java and C stack frames in the stack crawl window, all at the same time.

Dave: How will Java affect the world of web site management? Will CGI/Perl programming go away?

Marcus: One of the problems that people are starting to realize about Java is that you still need professional programmers to write applets. All Java does is increase the maximum power of expression available to web page creators; it does not make their task easier. JavaScript will likely have a greater impact on the use of CGI/Perl than Java. Also, web site management needs more powerful but simple-to-use tools. Adobe PageMill is a step in this direction, but there is a long way to go.

Clint: Perl programming will never go away. But the use of CGI/Perl solutions in web pages may dwindle as applets become easier to write, and as standard suites of applets become available to web page authors.

Dave: What is "just in time" compilation? Does Metrowerks support it?

Clint: JIT is on-the-fly compilation of Java bytecodes to native instructions, providing an enormous speed jump while not breaking the platform-neutrality of the binary. Our VM has hooks in it in order to support this.

Marcus: Marketing speak again: "Improving the execution speed of Java is an important future direction." Obviously,

the success of Java depends a great deal on its speed. The faster the interpreter, the more powerful and complex the programs that can be written.

Dave: How would you compare Java to other object programming languages you've worked with?

Tim: Because they've ripped out a lot of the "features" of C++ that cause problems, like pointers and direct memory manipulation, and have added features to make the programmer's life easier, like automatic garbage collection, I can see Java becoming a very popular development language. In any development, how many crashing bugs are the result of writing to the wrong piece of memory? That won't happen in Java: a big headache is gone. In addition, Java code is more readable and maintainable than most C++ code, because Java was written from the ground up as an object-oriented language, and has no feature compatibility to maintain with a cryptic language like C. All of the stuff that shouldn't be there isn't. The resulting code is clean and well organized, because Java pretty much has to be written that way. So bring on the Java-heads!

Clint: Java is a much cleaner language than C++, as it eliminates unsafe constructs like pointers, and provides automatic storage reclamation. It is statically typed (like C++), but is dynamically linked and loaded (like Smalltalk), thus providing for a much more loosely coupled language, which lends itself to a faster prototyping cycle. It has a whole slew of cool features, including typed exceptions and synchronization primitives. It also has a pretty complete set of language libraries. On the downside, Java does not support parametric polymorphism, and it does not provide for multiple inheritance of implementation. Overall, I consider it to be a very cool language that collects the better parts of a lot of existing languages. The best point in Java's favor is that it is designed to be a production language. And we're doing our best to help Mac programmers produce with it.

Marcus: I think Java has the right features for success. Its greatest trick is that on the surface it looks like C/C++, but is in fact a well designed object-oriented programming language. This means that all those C/C++ programmers out there who are biased against a properly designed language will use it because it seems to be a C++ derivative. Not used to looking deeply at the languages they use, they will be lured by the syntactic sugar of Java; beguiled by the surface similarities, they will become seduced by the garbage collection and type safety. Finally the world may start to use a "real" programming language and we can leave the dark ages behind.

MacRegistry™

Developer Job Opportunities

If you are a Macintosh developer, you should register with us! We have a database that enables us to let you know about job opportunities. When we are asked to do a search by a client company the database is the first place we go. There is no charge for registering. The database service is free. Geographic Coverage is nationwide.

Marketability Assessment - To get a specific feel for your marketability send a resumé via Email or call. You may also request a Resume Workbook & Career Planner.

Discreet - We are very careful to protect the confidentiality of a currently employed developer.

Scientific Placement is managed by graduate engineers, we enjoy a reputation for competent & professional job placement services and we are Mac fanatics.

1-800-231-5920 • das@spi.com • Fax 1-800-757-9003
http://www.scientific.com

Scientific Placement, Inc.

MT, Box 19949, Houston, TX 77224, 713-496-6100 Fax: 713-496-0373
MT, Box 71, San Ramon, CA 94583 510-733-6168 beth@spica.bdt.com
MT, Box 202676, Austin, TX 78720-2676 512-260-0123 lej@zilker.net
MT, Kenmore Station, Box 15225, Boston, MA 02215 617-424-8372 jen@spbos.pn.com
AppleLink: D1580; Compuserve: 71250,3001; eWorld: spi; AOL: davesmall

---

TRATTNER NETWORK

COMPUTER CONSULTING SERVICES

The Trattner Network (TTN) has the best opportunities for Macintosh developers in Northern California and Nationwide.

TTN represents clients with projects in Internet, Multimedia, Codewarrior and OpenDoc development among others.

Now you can visit us our web site at www.tratnet.com and see what openings are available for....

- **Software Developers**
- **Hardware/Firmware Engineers**
- **QA/QC Professionals**
- **Project Coordinator/Managers**
- **Multimedia Developers**
- **Network Professionals**

The Trattner Network has a unique history in **Mac consulting** coupled with exposure to emerging technologies. If you are looking for a chance to enhance your skills and marketability, **please send, fax, e-mail or link your resume to:**

The Trattner Network
Attn: Emily Hoolhorst
170 State Street, Suite 240 • Los Altos, CA 94022
Phone: 415•949•9555 ext.115 ; Fax: 415•949•1026
AppleLink: trat.net ; E-mail: emily@tratnet.com
http://www.tratnet.com

THE DIGITAL TALENT SOURCE

MACXPERTS

software design & engineering

"I can do anything with C++ and a good pair of boots."

– MacXperts Programmer approaching Himalayan village at 20,000 feet.

Adventurers Wanted

C, C++, Java, OpenDoc and other Macintosh, Newton, or Internet programming experience.

Call Michael Ruttle at:

1-800-356-8040 or fax / 804-358-3847
Internet: xperts@infi.net
AppleLink: xperts AOL: MacXperts
http://www.macxperts.com/~xperts

PARALLEL SOFTWARE

CHICAGO AREA

Parallel Software is looking for software engineers who are serious about component based **OPENDOC** development. We are interested in Macintosh or Windows people with the following experience:

"C" or "C++", Pascal, OPENDOC
Newton, NDK, or JAVA
MacApp,Code Warrior, MacToolbox
Oracle, Sybase, or SQL Databases

Please write or phone:

Mr. John D. McMahon
Parallel Software, Inc.
608 S. Washington, Ste. 101
Naperville, Illinois, 60540
Tel (708)369-0100
Fax: (708)355-7870
psgmcmahon@aol.com.

Do you have used equipment to sell?

Or do you want to purchase used equipment?

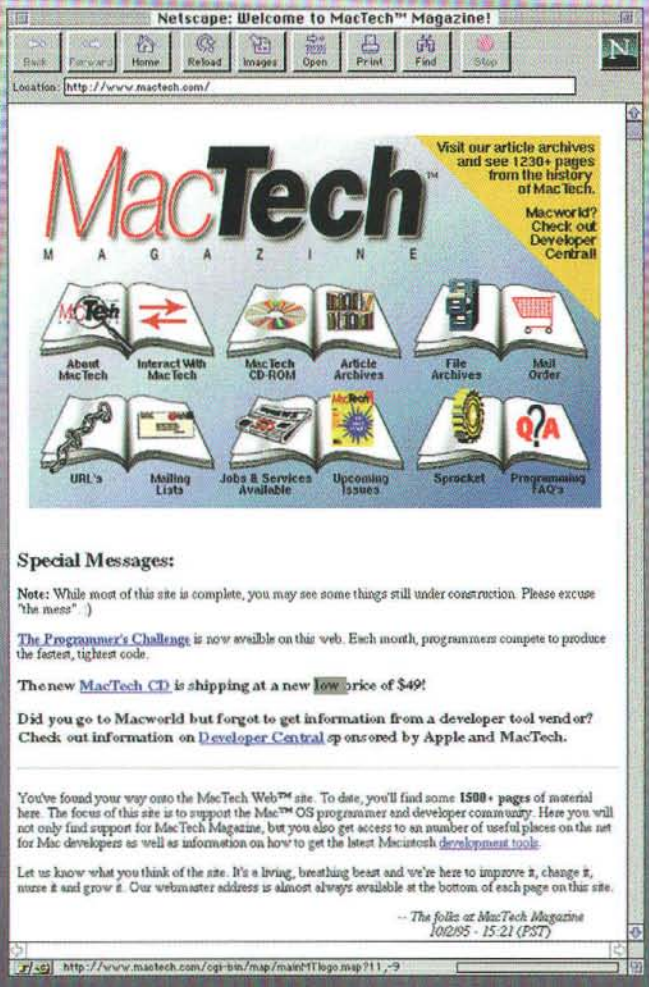
Advertise it in the Classified Section of MacTech Magazine. Call Ruth Subrin at 805/494-9797 to find out how to place your ad.

MacTech Magazine is your recruitment vehicle

When you need to fill important positions at your company, MacTech Magazine is the consistent choice of companies across the country for hiring the best qualified Macintosh programmers and developers. Let MacTech Magazine deliver your recruitment message to an audience of over 27,000 qualified computer professionals.

Call Ruth Subrin at 805/494-9797

http://www.mactech.com



(Need we say more?)

For Macintosh Programmers & Developers

MacTech™ MAGAZINE

CRABB'S APPLE

By Don Crabb

Software Updates and OpenDoc

How many of you are happy with your mechanisms for updating software for your customers?

For that matter, how many of you are happy with the mechanisms that Apple uses to update your customer's System software?

And what happens once OpenDoc makes an impact on your business? What happens when your own containers and your parts have to be updated? when other parts from other vendors that your customers will want to use with your containers have to be updated?

While we all might complain that the number of OpenDoc parts and containers is currently on the sparse side, we also know that this is going to change. Claris's plan to make the next version of ClarisWorks an OpenDoc container *par excellence* will drive the creation of a lot of parts. And you can expect other major non-Microsoft Mac vendors to follow suit. By the time of the first customer release of Copland, we're likely to be up to our keisters in OpenDoc. And the software update mechanisms that we and Apple have in place just aren't up to the task that this plethora of smallware will bring.

Today's Updates

How do you keep your customers up-to-date now? Let's consider their pros and cons:

- Mailings to those who send in their reg cards. You might get 50% of your installed base that way, if you are extraordinarily lucky and persevering and happen to sell a killer product. But that other 50% misses, and that ultimately costs you money.

- A Web/FTP site. This works well for your more advanced customers, especially corporate or higher ed customers with the high-speed Net connections needed to suck down updates regularly and the IS staff to make sure they get paid for and redistributed among their Mac users. Web sites work less well if you have a substantial SOHO or K-12 user base, as they often don't have the time or money or Net expertise to make the daily Net connections needed.

- Dealers. This can work if you mostly sell through large computer or consumer electronics superstores and you only expect to issue major upgrades. Otherwise, it's expensive and doesn't have a lot of reach.

- Mail-order dealers. Pretty much the same advantages and caveats hold as for large non-mail-order dealers. With catalog space at a premium for mail-order dealers, it's likely to get even more expensive to update parts that way.

Keeping Track Now

Your customers, of course, can avail themselves of other sources of information about your latest wares (as well as Apple's) and how to get them. One of the best of these sources is Level 6 Computing's monthly software update report. It costs $150 per year ($97 per year for independent consultants) and comes as a comprehensive 24-page printed report, a disk with vendor contact information on it in setext format (along with clickable FTP sites and URLs), and a Web site (www.webcom.com/level6/). You can contact them at update@level6.com. If you don't send your update information to Level6, I urge you to do so.

Posting information to the various Apple newsgroups (Guy Kawasaki's Semper Fi and MacWay lists of course, as well many others), to TidBITS, and to the Info-Mac lists is also a must do, as well as notifying the usual suspects – *MacWEEK*, *MacUSER*, *MacWorld*, and *Mac Home Journal* among others.

But even when you use all of these methods for keeping your customers updated, even when you make multiple methods available to them for obtaining those updates, the simple truth is that you end up missing a lot of them. And the smaller you are, the more of a problem this is – it may even mean the difference between success and failure. If you doubt this, just look at the latest reports from *The Hartsook Letter*, InfoCorp, and other Apple-tracking agencies – these show that a small but nontrivial portion of Mac customers are lost each year simply because they did not know how to get software updates and assumed that updates were just no longer available on the Mac platform – so they moved to Windows to get the latest version.

Continued on page 91

NEWSBITS

By John Kawakami

DataPak Reveals the Secret of PAIGE's Success

PAIGE, DataPak Software's (DSI) popular cross-platform text engine, is scheduled for a major 2.0 upgrade.

To determine what type of functionality was missing from the current version, DSI's marketing department interviewed and surveyed hundreds of clients and prospects. "The 2.0 upgrade is a major step for the technology, and it was very important that our clients were involved in that process. After all, they are the ones putting the technology through its paces," stated Mark Nulph, Director of Marketing for DSI. "Besides, it's tough for us to anticipate the needs of our clients. Let's face it, clients like CompuServe, America Online and Prodigy are going to have different text considerations than, say, Macromedia, Scitex, Caere or Grolier Electronic Publishing."

Source code clients will have the first look, and the final version is scheduled for broad release some time in May. The following is a partial list of suggested features/enhancements expected to be in the PAIGE 2.0 upgrade:

- Built-in RTF Import/Export
- Full MFC support (including a new PAIGE demo written entirely with MFC)
- OLE support
- Support for Drag and Drop
- Sub-Containers (makes creating forms easier)
- Unicode Compatibility
- Electronic documentation
- OpenDoc Support
- 2.0 will employ many open standards to ensure inter-operability. Under consideration are CORBA, ODBC, Bento, SGML, HTML, SOM, Java, etc.

DataPak Software Inc. 11815 NE 99th St., Suite 1200, Vancouver, WA 98682. Sales: (800) 327-6703. Email: sales@datapak.com.

http://www.datapak.com/~datapak/

Three Dimensional Machination

Apple, Netscape, and Silicon Graphics International announced that they are working together to use the QuickDraw 3D metafile format as the binary format for Moving Worlds. Moving Worlds is SGI's proposed new Virtual Reality Modeling Language (VRML) standard, which, at press time, hasn't yet been picked as the new VRML, but has significant computer industry support. Netscape, SGI, and now Apple, together present a formidable challenge to the competing standards.

Previously, Apple had its own proposed standard competing with Moving Worlds, but changed direction and refocused its efforts toward promoting the QD3D metafile after SGI's proposal garnered the support of dozens of other companies. Apple's promotional efforts, if successful, will help bridge the gap that exists between QD3D and VRML, and improve QD3D's position in the market. It would also increase Apple's ability to influence the VRML standard.

Apple's contribution of 3DMF technology to the Moving Worlds effort will enable higher compression rates, file streaming, and faster parsing of 3D objects and virtual worlds across the Internet.

http://www.info.apple.com/qd3d/

Royal Acquires Heizer

Royal Software, Inc., and Heizer Software jointly announced that Royal Software will acquire Heizer General Corp. for an undisclosed amount. Heizer has long been the leading publisher of CompileIt! and WindowScript, tools for use with HyperCard and other authoring environments. Royal Software, conversely, is an unknown entity led by long-time Heizer client Ro Nagey, who's famous for sponsoring trips to the former Soviet Union to fly Russian jet fighters. Royal Software will continue to use the Heizer Software name, and plans to pursue online marketing and additional products aimed at entry-level HyperTalk users.

Heizer Software, A Division of Royal Software, Inc. 300 Cedar Lane, Largo, FL 34640. Phone: (813) 581-6422. Email: ronagey@heizer.com.

http://www.interedu.com/heizer/
http://www.interedu.com/mig29/

Here's Nutmeg For Your Java
Can Steamed Milk Be Far Behind?

THOUGHT Inc. announced Nutmeg, one of the first commercial class libraries available for the Java Language.

Even though it is one of the most promising new technologies to arrive, Java programming can still require a great deal of time for low-level programming tasks. One of the areas where this is true is in list management, which is still fairly primitive in the Java Class libraries. Smalltalk has long been known as a powerful programming language for applications

like list management. THOUGHT Inc. has sought to address the problem of making Java useful for low-level programming tasks by creating a set of Java classes entitled Nutmeg, which have been modeled after the Smalltalk Collection Class library. Nutmeg for Java is one more tool to assist programmers in developing advanced applications using Java.

Nutmeg features include classes to manage: arrays of information, indexed collections, ordered and sorted collections, sets, and even dictionaries. Users can also define a Smalltalk-style or Java-style of error handling (exception handling), as well as programmer-defined sorting criteria and, in many areas, programmer-defined error handling and object selection blocks. These more robust classes should greatly improve Java's applicability for large-scale and serious development.

A limited evaluation copy of Nutmeg is available for download under "products" on their Web site. Corporate pricing is expected to be US$495 per seat for the binary release, US$995 per seat for a source code license. Student pricing is US$49 for the binary. [*Time to register for classes! – jk*] These classes have been designed to work with either the Windows, OS2, Macintosh or Unix based Java Developers Kits.

http://www.thoughtinc.com/

There's a Company Behind ftp.pht.com!

Pacific Hi-Tech, which has, for some time, mirrored the Info-Mac archives at ftp.pht.com, recently sent us a notice regarding three CD ROMs:

MacSource is a disk containing 600 Megabytes of source code, compilers and languages. 200 Megs of this is source code, 150 of C/C++, 25 of Pascal, and the rest BASIC, Lisp, Assembly, Applescript, Prolog, and others.

Hyperstacks is a disk containing more than a thousand freeware and shareware stacks. [*Including several classics by Your Humble Managing Editor. – man*]

Internet PowerWEB is a book/CD-ROM combo intended to assist in the writing of Web pages. The book is a tutorial in three languages, and the CD-ROM is all of the most popular Web writing sites on the Internet.

http://www.pht.com/

Oracle Plans for The Future

Oracle has been expanding into microcomputer software development tools, and has been focusing on the Macintosh as one of its key platforms. Not only do they produce high-powered cross-platform development environments for the Macintosh, they even make an effort, in their marketing material, to appeal to Macintosh developers. To wit, an Oracle executive commented about their Oracle Media Objects development tool:

"Oracle Media Objects 1.0.5 adds a Windows 3.1 development environment to the already available Macintosh development environment. Players for Macintosh and Windows are included with both development versions. OMO is fun and easy to use, and gets your applications up and running quickly. Developers can easily extend the functionality of OMO using the MOX (Media Objects eXtensions) interface. The OMO metaphor is based on the card and stack, just like HyperCard."

[*When I tried out the demo, it was a lot like HyperCard, but with many new tools. It was much slower that I expected on a Power Macintosh. Perhaps there's some significant overhead incurred by running under Oracle's cross-platform environment. – jk*]

You can download the 17 megabyte trial of OMO at the Oracle Web.

Oracle Web
http://www.oracle.com/

Oracle Media Objects Web
http://omo.us.oracle.com/

Peregrine Shapes Up to Challenge Client/Server Tools

Peregrine is a rapid application development tool for creating compiled client/server applications and databases. Peregrine is completely visual and object-oriented so code is reusable and extensible. It has a built-in database for testing and prototyping applications without overloading the server, as well as for creating standalone database applications. Peregrine provides true client-server architecture and includes full source code, giving developers control right down to the Mac Toolbox level.

Peregrine 1.3.2 features:

- ODBC driver support, which increases the number of databases which can be connected to simultaneously and in the same application. Other databases supported by high performance drivers are Oracle, Sybase, Microsoft SQL Server, Butler, dtF and DAL/DAM.
- A Power Mac compiler for compiling Power Mac native client applications.
- Support for dtF LAN.
- Peregrine Xtras - including 4D Migrator Tool for moving schemas and data from 4D Server into Peregrine's internal database or to another Peregrine-supported database. The Xtras also include an Application Generator and window items that ease the creation of relationships.
- Preview of the Windows version of Prograph CPX, the underlying programming language for Peregrine.

Pictorius, Inc. 2000 Barrington Street, 4th Floor, Halifax Nova Scotia, B3J 3K1, Canada. Sales: (800) 927-4847. Email: info@pictorius.com

http://www.pictorius.com/

The Claris World Wide Developer's Conference (Not to Be Confused With Apple's WWDC)

Claris Corporation is holding its first-ever Worldwide Developer's Conference (WWDC) June 3–5, 1996, at the Santa Clara Marriott Hotel in Santa Clara, CA, and will offer both in-house and third-party developers hands-on sessions using the hottest Macintosh and Windows technologies with Claris and Apple software.

"Developers have become increasingly successful deploying solutions based on FileMaker Pro and ClarisWorks across small business, education and workgroup environments," said Guerrino De Luca, president of Claris Corp. "With technologies such as OpenDoc coming to the forefront, and the Internet emerging as a platform, Claris more than ever needs to listen to our developers, support them, and give them the tools they need to deliver custom solutions and services based on our products."

The first Claris WWDC will include three days of developer-specific sessions for Macintosh and Windows developers, with the objective of providing tools and techniques to maximize solutions based on Claris products, and will cover several topics, including:

- Deploying database solutions over the Internet using FileMaker Pro.
- Designing user interfaces for Macintosh and Windows solutions.
- Integrating OpenDoc and OLE with Claris applications.
- SQL connectivity with FileMaker Pro and third-party programs.
- Marketing and selling solutions and services.

Developers interested in getting registration information, or who have other Claris WWDC questions, can call (800) 778-9383 (North America) or (503) 699-7178 (International) for more information.

DIALOGUE BOX

By Matt Neuburg, letters@mactech.com

If They Build It, Will Anyone Come?

We thought that readers might be interested in these excerpts from a thread that occurred a while back (September, 1995) on the Apple internet providers mailing list (subscription info at http://www.solutions.apple.com/apple-internet/*) between ZiyaOz@aol.com (Ziya Oz) and jon@comvista.com (MacTech Magazine contributing editor Jon Wiederspan). Obviously the details of the situation have changed in the meantime on many of these products (please bear this in mind while reading!), but the point remains valid, the more so since, as this issue went to press, Apple was actively soliciting internet strategy advice from users.*

Ziya: Are you bothered by the alarming absence of serious HTTP servers (as well as site-management, clients/browsers and visual authoring tools) for the MacOS? I've just begun to compile a list of all significant servers, clients and tools available for Windows NT and Unix, but *not* for the Mac. The objective is, first, to document the severity of the situation (so that perhaps Apple can put it on its radar screen) and, second, to pressure some of the companies that completely ignore the Mac, to pay more attention to it.

PRODUCT	CLIENT/Browser	SERVER	DEV TOOL
OpenUI	Q4 95	No	No
WebObjects/Next-OpenStep	No	No	No
Netscape/LiveWire	Yes	No	No
Verity Topic Agent/Server	Soon?	No	No
ATT Interchange	No	No	No
Microsoft Network/Blackbird	No	No	No
Hyper-G	Soon?	No	No
Vermeer FrontPage	Q1 96	No	No
RAD PowerMedia	Soon?	No	No
SapphireWeb	No	No	No
ArchiText	Yes	No	No
Sun Java	Soon?	NA	Roaster
W3 Website Toolkit	No	No	No
WebBase (ODBC)	No	No	No
NaviSoft/NaviServer	Yes	No	No
OpenMarket	No	No	No
O'Reilly WebSite	No	No	No
InContext Spider	No	No	No
Microft Explorer	No	No	No
Wollongong Emissary	No	No	No

Jon: I'm a bit confused by the point of your list. Do you really mean that every server that runs on another platform should also run on a Mac?

Ziya: This is not just about HTTP servers. It's also about developer tools, browsers, site managers, editors, etc. It's really about what I called an alarming trend on the Mac. Here's what I mean:

At the last MacWorld Expo in Boston, I asked myself: If Ceneca was not formed by ex-Taligent people, would they have developed Page/SiteMill for the MacOS first? Would they even have developed for the Mac at all? After seeing at the Fall Internet/Boston last week the preponderance of Windows NT and UNIX, and the widespread neglect of the Mac, my answers are: No and No again.

By now it's futile to cling to an unreasonable hope that the Mac's role in corporate "business" computing (databases, enterprise apps, servers, CAD, network management, etc.) won't become less and less consequential. Instead of engaging in quasi-religious discourse on this, let's move on. Is there a future for the Mac in other areas? Certainly, in education, 3D, digital video, graphics, DTP, interactive authoring and, I'd have thought, Internet.

I watched Spindler tell the audience at the launching of the PowerMacs that "Apple would do Internet right." Two years later, Apple has precious little to show for it, with the possible exception of the promise (and no more) of Cyberdog. How else could it be when (as I'm told by the OpenDoc evangelists) there's no person(s) in charge of Apple's Internet strategy, product and marketing? Of course Cyberdog is a great concept, but so were PowerTalk, QuickDraw GX, GeoPort and a host of others Apple introduced in the recent past. Is Internet important? A lot of people think so. Is it important, or even vital, to Apple? How can you tell?

Internet is essentially a client/server architecture: browsers request and servers serve data of some kind. Apple is not in the browser business; that's owned by Netscape (and soon to be shared by Microsoft). Cyberdog can't really change that in a 90% non-Mac marketplace. Well then, is Apple in the HTTP server business? Not really. The only commercial HTTP server of any size is WebStar. (InterServe, Web Server 4D, Netwings, etc., may change that. We'll see.)

It's admittedly difficult to make a great and attractive server. When your share of the OS market is 9%, though, you cannot afford not to be really great. Currently, lack of true

multitasking, multithreading, multihoming and myriad file and bus I/O issues cripple the Mac against many others in the NT and UNIX world. OpenTransport may alleviate some of these problems (and Copland others), but NT and UNIX don't have these problems *now*, and they sure won't remain stagnant either.

If you were a Windows user, say, two years ago and went to the Seybold show (for the digital graphics, DTP and prepress industry), you were alienated and felt virtually ignored. Sure, you had PageMaker and CorelDraw for Windows, but all the exciting technology came out on the Mac and developers concentrated on its advantages. Yet when I went through the Internet World last week, and stopped by nearly 100 booths, the overwhelming majority of them made me (a Mac developer) feel, well, irrelevant.

That to me means that those who develop for and profit from Internet are finding the Mac not so relevant. I've also talked to countless "corporate" types who are searching for tools to publish internal data and conduct commerce on a large scale. Uniformly, they did not consider the Mac as a viable platform for the server/backend operations. In the end, business people are pragmatic. If your product does not offer demonstrable superiority, they'll ignore it. (Mac is still superior to Win 95, NT and UNIX in DTP and graphics, so it's still favored.) Since the Mac does not offer sufficient unambiguous advantage for the Internet, unless something drastic happens soon, it will eventually be ignored.

When the browser used by 80% of Internet announces that it will not release its servers, visual HTML editors or scripting tools for the Mac, you have to wonder why and worry some. When a large number of people start developing with Blackbird and you cannot even use it on the Mac, you don't need to wonder but you'll have to worry a little.

If your business needs to access AT&T Interchange or Dow Jones Personal Journal on-line, or conduct sophisticated commerce with an Open Market server, or integrate your internal Oracle or Sybase databases with Web access via object-oriented tools such as WebObjects, or if you want to serve disk- and CD-ROM-based search engines like the Verity Topic Servers – well, you are out of luck if you are a Mac shop.

And if the Mac is not a serious player at this early stage when tools are simpler, what happens to more sophisticated Mac technologies like QuickDraw, QD 3D, QT VR, etc., in terms of future acceptance? You think it'll get any easier? When increasingly more sophisticated plug-ins via Navigator talk to increasingly more capable and varied servers that don't and won't, apparently, exist on the Mac, how does that help the Mac user? If a developer of an HTTP browser or server plug-in has the impression that there are only a few servers on the Mac and, more importantly, throughput of the Mac servers is limited compared to, say, an NT box costing about the same as a PowerMac, do you think he will worry too much about that 9% Mac market share?

OpenDoc Solves the Wrong Problem

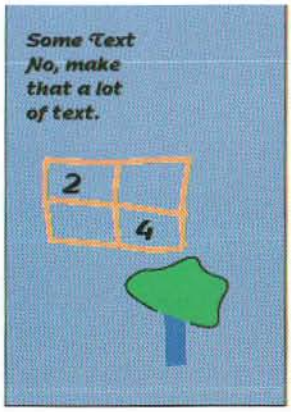
I've been worried about OpenDoc. My worry is that Apple and its allies are spending a lot of time working on a technology which is ultimately going to turn out to be irrelevant to users. A columnist in *PC Magazine* pointed out that compound-document technologies seem oriented towards enhancing the applications of today, rather than those of tomorrow. Then, last night, my misgivings crystallized at last, in a form that I think I can explain coherently.

Consider a sheet of paper. Do I hear someone ask, "Do you mean a word-processing sheet of paper, or a graphics sheet of paper?" Of course not! Stupid question. A sheet of paper is a sheet of paper. Contrast this with the modality of most current software: before you create a new document on your Mac, you must first decide whether it's going to be a word-processing document or a graphics document. (ClarisWorks fans should not start smirking just yet.)

The essence of pre-OpenDoc software: *an application is a mode.* Remember, a mode is a state which is not quite the one you want to be in to do what you want to do next, and you have to consciously think about switching to the right state. Having different states for your software is not evil in itself – the secret is to match them to the problem at hand, and to make it as easy and natural as possible for the user to move between states, without even thinking about it. When a state becomes restrictive and unnatural to the problem at hand, then it becomes a mode.

Back to that sheet of paper. I pick up a pencil, and write some words in one part (a word-processing part!), draw a table with some numbers in another part (a spreadsheet part!) and put a graphic in another part (a graphics part!).

So far, so OpenDoc. Now, just for fun, I want to put a graphic into one of the table cells. Will OpenDoc allow me to do this – use the tools of one part to work on another part? No it won't. Whereas, on a sheet of paper, every tool (pencil, pen, charcoal, eraser, paintbrush, typewriter, finger, whatever) works in some way on every part.



Hence, the essence of OpenDoc software: *a part is a mode.* Sure, a spreadsheet part could allow embedding of other

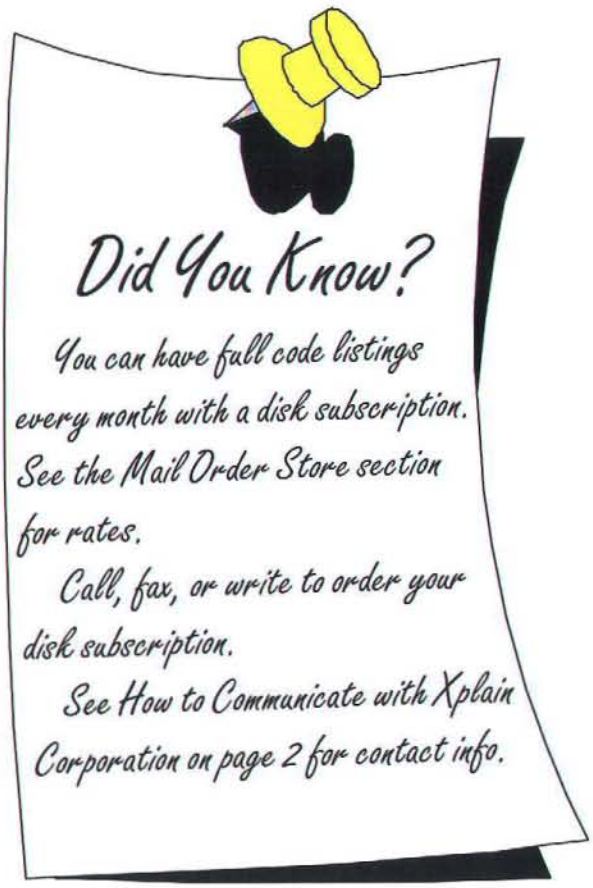
parts within its cells, but you can appreciate that this only gets you a little closer to the modelessness of a sheet of paper.

So how should we approach the problem instead? The beauty of a sheet of paper is not just the variety of ways in which you can make marks on it, but that these different ways can interoperate – a mark made by a paintbrush can freely abut or even overlap a mark made by a pencil, unlike OpenDoc, with its demarcation of the document into parts. If you could make the granularity of parts fine enough, you might manage it, but I don't think the part idea is set up to deal efficiently with a page containing hundreds of parts.

In short, the problem needs to be rethought in terms of the basic technology for putting marks on the computer screen. In order to achieve true, paper-style modelessness, different tools need to be able to operate on a common graphical representation of their data.

OpenDoc is a step in the right direction, in the sense that building a very high tower is a step towards getting into outer space – in short, it's a step that I don't think is worth taking. I think the effort would be better expended on solving a larger chunk of the problem, on the rockets that will take us closer to true modelessness.

Lawrence D'Oliveiro, ldo@waikato.ac.nz



Crabb's Apple continued from page 85

Future Updates

Web design and technology will get better, as will the availability of the high-speed networking (ISDN, cable modems, ATM) necessary to bring all your customers to the Web update trough. But even with slicker Web sites and better navigation aids to find them, we still need a breakthrough in the way that we'll deliver updates to our customers without them having to even think about it.

That breakthrough – simply put – needs two things. One, we and Apple have to simplify our upgrades. Just this past week I received update disks and several CD-ROMs for a dozen different Mac products I use anywhere from daily to once in a blue moon. But from the packaging these updates came in I hadn't a clue which I should apply immediately and which I should dump. Only after reading all the paper stuff in the packages, the ReadMe files, and taking a good look at the files and installer provided, did I have a clue as to the importance of each update. And I do this sort of thing for a living. I don't need to imagine how hard it is for others whose real jobs are *not* bit-twiddling, but use their Macs to get their *work* done. I don't need to imagine, because I get a bunch of calls each day from these folks asking me what the hell they should do with the SuperWhizBangPro V.3 Updater 8.12 they just got in the mail. (And don't give me that stuff about the version number meaning anything. I've had a release 1.2.3.4 that was critical to my operation and a release 2 that was a total waste of time.)

And two – we need a mechanism for better communicating the real point of an update (and why customers should pay for it, if it's not a freebie).

Only after we've done both of these can we focus on getting it to customers in a painless and trivial way. My guess is that we need to look at the whole process as a continuum and not as a set of discrete problems. Rather than divorce the installer from the Web/FTP download widget from the email that was sent to the customer to start the process, we need to figure a way to build the three parts together as an "Update Object" that gets sent to the customer (via email, via a disk or CD-ROM in the snail-mail, etc.) every time we issue an update.

This "Update Object" would have the mechanism for obtaining the full update from its Net site, along with the chargeback method, and a clean, reasonable explanation of why your customers need it and what it does. Once they say yes, a simple button click would do all the rest. Pay for it. Download it. Scan their system for its compatibility. Set the proper installation options. Do the installation. Then test that installation went correctly (restarting, etc.), while fully informing the customer on their screen what was taking place. For the wireheads among your customers, the "Update Object" could also provide various manual interrupts or single-step action.

I suspect that if any of you think about this problem for a bit, you could architect the shell for this "Update Object". I also imagine that if you license it to your fellow developers (and to Apple), you'll win the gratitude of their customers, as well as yours (and you'll make a few bucks along the way).

TIPS & TIDBITS

By Steve Sisak

TIP OF THE MONTH

Invoking Handlers in Scripts, by Name

This is completely undocumented, as far as I know, but it may help. Suppose you have a script application containing a function:

```
on DoSomething(param1,param2)
  return param1+param2
end DoSomething
```

From C, you just have to send the script application an Apple event like this one:

CLASS: `ascr`
type: `psbr`
direct object: `"----" "LIST"`
(the list of AEDescs of parameters to pass to the script function)
additional parameter: `"snam" "TEXT"`
containing the name of the function to call (in our example: "DoSomething")
et voilà.

This method allows to write "clean" scripts, using more memorable handler names, not using the `«CLASStype»` syntax.

Pierre-Loic Raynaud

[This event is called "Subroutine Call" and is described in more detail in Chapter 10 of the Apple Event Registry, "The AppleScript Suite". On the Developer Mailing Reference Library CD, the pdf file is named "AppleScript Suite", and the information is on pdf page 5 (paper page 7). – jk]

Swapping Bytes in a High Level Language, the Saga Continues!

You have probably been swamped with everyone's comments regarding the "Anti-Tip of the Month" that appeared in *MacTech Magazine* 11.10 (October 1995).

Greg Poole had the right idea when he submitted his tip about byte-swapping. You can tweak code until your fingers fall off, but often the best way to make something faster is by finding a better way of doing the same thing.

I have attached two files to this message: `ByteSwap.c` & `ByteSwap.h`. In a nutshell, we do our swapping as follows:

ByteSwap.h

```
#define SwapShort(myUnsignedShort)  \
            ((myUnsignedShort)>>8)|((myUnsignedShort)<<8)
```

In use:

```
{
    unsigned short someValue = 0x3210;
    someValue = SwapShort(someValue);
}
```

We quite simply move the hi-byte right, and the lo-byte left, then OR them back together.

ByteSwap.c

```
#include "byteswap.h"

unsigned long TransposeLong(unsigned long value)
{
```

*Send us your tips or we'll install EvenBetterBusError on your machine! On the other hand, we might just pay you **$25** for each tip we use, or **$50** for Tip of the Month. You can take your award in goods, subscriptions or US$. Make sure any code compiles, and send tips (and where to mail your winnings) to our **new Tips e-mail address** at tips@mactech.com. See page two for our other addresses.*

```
    unsigned long   returnValue;
    ((unsigned char *) &returnValue)[3] =
                            ((unsigned char *) &value)[0];
    ((unsigned char *) &returnValue)[2] =
                            ((unsigned char *) &value)[1];
    ((unsigned char *) &returnValue)[1] =
                            ((unsigned char *) &value)[2];
    ((unsigned char *) &returnValue)[0] =
                            ((unsigned char *) &value)[3];
    return returnValue;
}
```

David Most

The Technical Editor Responds

There are quite a few methods available for byte swapping, but which algorithm is the best depends a great deal on the compiler and processor you are using. For instance, the PowerPC (and many other processors) have instructions specifically for this purpose. Unfortunately, since the C and C++ languages have not kept up with current processors (C is basically processor-independent PDP-11 assembly language), it is our job to trick the compiler into generating the correct code.

Some compilers provide directives for this purpose. For instance, if you are using CodeWarrior on a PowerPC you can just say:

```
inline long SwapLong(long val)
{
    return __XXXXX(val);    // <<<need intrinsic for load byte-swapped>>>
}
```

You may notice that I'm using C++ inline functions here instead of C `#defines`. To quote the "Apple Unofficial C++ Style Guide" (**develop** 2, p. 209): "One of the most powerful features of the C++ language is the C preprocessor. Don't use it." Inline functions are not only more readable than preprocessor macros, but, because they limit side effects, allow the compiler more latitude in optimizing your code.

Barring that, we need to find something that we can say in C that can get bytes swapped without generating egregiously bad code. David's solution is probably a good one for shorts because it is a pure mathematical expression, allowing the compiler to optimize it in any way it chooses. For longs, his solution is one of the safest, if you know nothing about the compiler and/or processor you're building for. (I would, however, convert both to inline functions.)

However, if you do know something about your processor, you can do better. In the case of the PPC, you really want to get your compiler to emit a load byte-swapped instruction. On a 680x0, you have a little latitude. One trick that comes to mind is that the 680x0 has predecrement and postincrement addressing modes. This means that:

```
foo = *p++ and foo = *--p
```

are fast and (1 instruction)

```
foo = *++p and foo = *p--
```

are slow (3 instructions). Therefore we can swap a long on a 680x0 with:

```
inline long SwapLong(long val)
{
    Byte* p = ((Byte*) val)[4];    // 680x0's are big-endian

        long val = *--p;
        val  = (val << 8) | *--p;
        val  = (val << 8) | *--p;
       return (val << 8) | *--p;
}
```

The PowerPC has only post-increment instructions, so this will generate lousy code. (If someone would like to time a bunch of approaches, I'd be glad to publish the results.)

I must say that the bottom line is: this is all a bunch of work that would be completely unnecessary if the C language had kept up with reality. Here's my proposal to the ANSI committee:

Make `littleendian` and `bigendian` storage classifiers like `const` and `volatile`. Then I could just type:

```
typedef struct Foo
{
    littleendian long    bar;
    bigendian short      baz;
} Foo;

void blah(Foo* foo)
{
    long  bar = foo->bar;
    short baz = foo->baz;

    ...
}
```

...and let the compiler deal with it while I spend my time writing code which does real work.

– sgs

Fix For Tip Of The Month, January 1996

I'm sending this short note just to point out that, although Greg Poole is right in writing that a file or directory can be moved by the `CatMove` function only if both the source and destination are on the same volume, he seems to forget that every Macintosh volume, not just the System volume, has a Trash folder.

Thus, if we pass to the FindFolder function the volume reference number of the file to be deleted instead of the constant `kOnSystemDisk`, we will be able to find the directory ID of the local Trash.

Line #44 of "FSpTrashFile.c" source file should be changed from:

```
theErr = FindFolder( kOnSystemDisk, kTrashFolderType,
                     kDontCreateFolder, &vRefNum, &dirID );
```

to:

```
theErr = FindFolder( (*theFile).vRefNum, kTrashFolderType,
                     kDontCreateFolder, &vRefNum, &dirID );
```

Live Long and Prosper!

Luigi Belverato

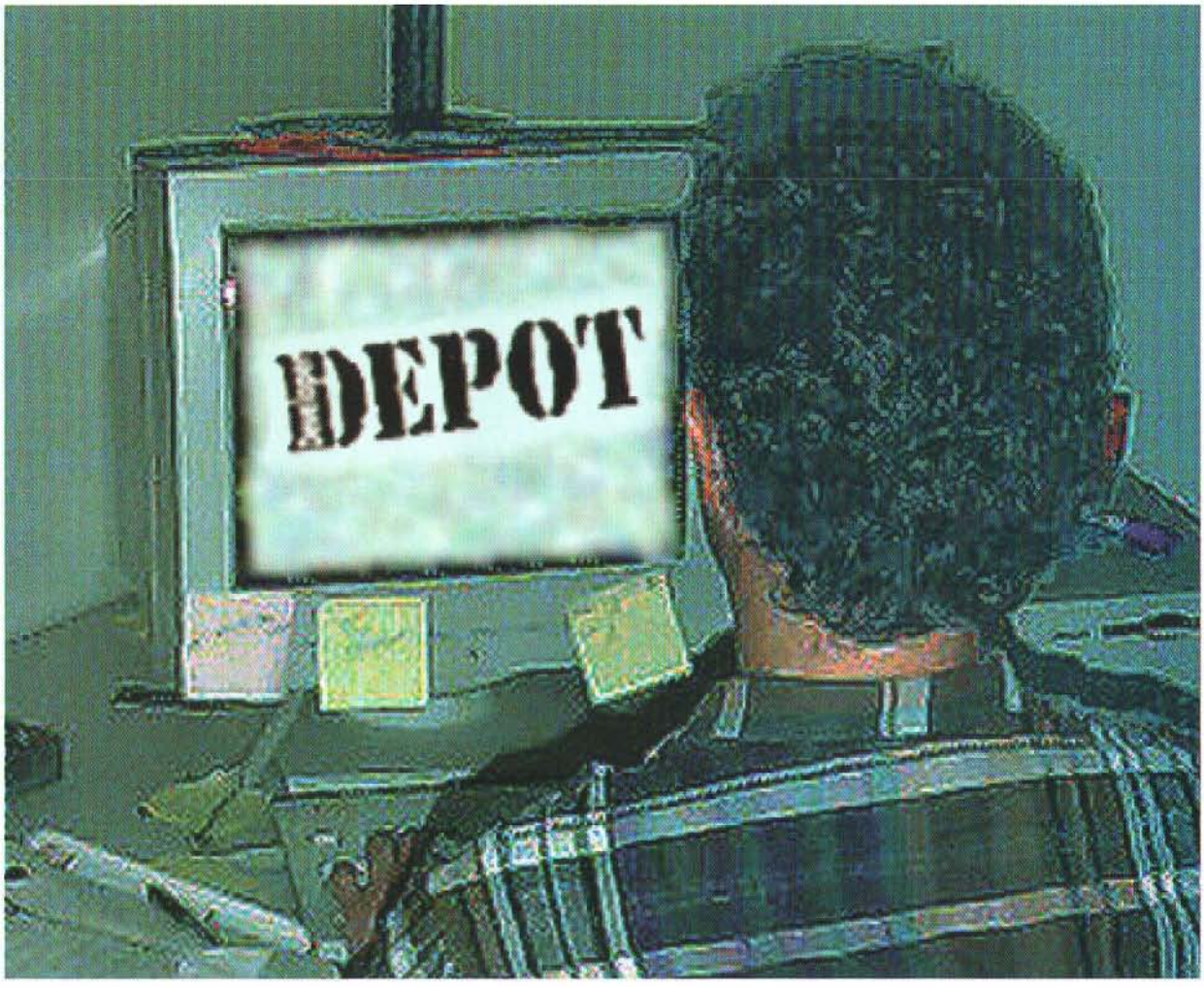
ADVERTISER/ PRODUCT LIST

LIST OF ADVERTISERS

LIST OF PRODUCTS


Mac OS
Scripter
The ultimate authoring and development environment for AppleScript
DANNY GOODMAN'S
Apple Guide
Starter Kit
Danny Goodman
Jeremy Joan Hewes
FRAGMEN
OF YOUR
IMAGINAT
Code Fragme
Resources for Pow
and Macintosh
Joe
SIC for the
ewton
John Schettino · Liz O'Hara
New Technology Building Blocks
Macintosh
Programming
Techniques
A Foundation for All Macintosh Programmers
Dan Parks Sydow
LEARN
C
ON THE
MACINTOSH
SECOND EDITION
SUBSCRIPTION
VERSION
SYMANTEC.
C++
DEVELOPMENT SYSTEM FOR
POWER MACINTOSH
EWARRIOR
RONZE 8
CodeWarrior
FACESPAN
The Interface Designer and Application Builder.
Dave Mark Cartwright Reed
Macintosh C
Programming
PRIMER
Volume I
Second Edition
Inside the Toolbox Using THINK C
OPTIMI
Power
CODE
Gary Kacmar
leScript
Additions Guide
Essential
Developer
DEPOT

Open Programmer's Hours...



...and all day too. In fact, we're open 24 hours a day, 365 days a year!

Which means that whenever you get a craving for the latest in developer tools, we're just a couple of keystrokes away.

You can browse for hours (we don't mind), pick up the items you want, and conveniently pay with your major credit card at our virtual check-out. Then, kick back and wait for your goodies to arrive - all without ever leaving your house!

Best of all you'll shop confidently, because at Developer Depot, you're guaranteed complete satisfaction and the best prices for 30 days after your purchase.

Talk about a safety net!

That's right, and speaking of nets, get connected and come check out the vastest on-line selection of developer products... any day, any time!

Developer DEPOT

http://www.devdepot.com

Dear Developer,

Since March, 1985, **MacTech**™ Magazine (then MacTutor) has been the place for you to find Macintosh Development products – the majors as well as those harder to find. Back then … and today, our focus has always been for the Macintosh developer community to grow and prosper.

But you deserve more than simple listings – and you've asked to make it easier to find and order products while maintaining our renowned customer service and pricing. We've been listening …

We are therefore proud to announce **Developer Depot**™ – a new mail order concept and entity dedicated to you, the Mac OS Developer. What makes Developer Depot so good?

- World renowned customer service
- 24-hour and full accessibility via our continually updated web site (http://www.devdepot.com)
- Satisfaction and best price guaranteed
- A new, easy to remember toll free phone number (**800-MACDEV-1**)
- Great selection (hundreds of products)
- Convenience: you can order in many ways (web site, e-mail, fax, phone)
- A new full color, organized, and expanded catalog – updated monthly

Additionally, we are a Macintosh house using an all Macintosh system. Our knowledgeable staff can answer your questions about products – and if they can't, they'll find the answers for you.

While Developer Depot is completely separate from MacTech Magazine, Developer Depot is the primary source for all MacTech products.

Let us know what you think – we're here to serve you.
Developer Depot: The products you need with the service and prices you deserve.

Neil Ticktin
Chief Executive Officer

Andrea J. Sniderman
Chief Operating Officer

TABLE OF CONTENTS

Developer Depot 30 day Money Back, Price and Satisfaction Guarantee
Developer Depot products are sold with a 30 Day money back guarantee on user satisfaction, lower prices and against defects. If, for any reason, you are not satisfied or find the same product at a lower price within 30 days, please call Customer Service at 800-MACDEV-1 and request a Return Merchandise Authorization (RMA) number to get a full refund or the difference in price (where applicable). You must return undamaged product at your expense, including all its original packaging, documentation and the blank warranty card if applicable. Developer Depot will replace defective product upon receipt of the defective merchandise. Please remember to back up your data before installation of any new hardware, software, or peripherals; we cannot be responsible for any lost data. Policies, item availability, and prices are subject to change without notice. The price in effect when we receive your order will be the price that you are charged. We are not responsible for any typographical errors in this or any other catalog, nor for any misstatements from any vendor. Purchase orders are also accepted but must be in writing, signed, come with a contact person and a telephone number, and mailed to P.O. Box 5200, Westlake Village, CA 91359-5200. Faxed copies and purchase order numbers alone are not acceptable.

Stuff our lawyer made us write.
Developer Depot makes no other warranties. All other warranties, either expressed or implied, including the implied warranty of merchantability and fitness for a particular purpose are disclaimed. Developer Depot shall not be liable for any direct, special, incidental or consequential damages including lost profits, from any delay in delivery, or for any personal injury arising from the use of any product sold through Developer Depot. The limit of direct damages, if any, shall not exceed the purchase price of the product.

For Macintosh Programmers & Developers

MacTech™ MAGAZINE

MacTech Magazine

Subscriptions: US magazine: **$47** for 12 issues
Canadian: **$59** for 12 issues
International: **$97** for 12 issues
Back Issues: **$5** each (subject to availability)

MacTech CD-ROM Volumes 1-10

- Over 1230 articles from all 115 issues (1984-1994) of MacTech (formerly MacTutor magazine)
- Includes all article text and source code
- THINK Reference format – for lightning quick access!
- Also includes: Symantec's THINK™ Reference 2.0, working applications with full documentation, product demos for developers, and much more!

List $89.00 Our Price **$49.00**

Upgrades: Our Price **$39.00**

MacTech Mouse Pad

Slide on this! With an extra-large surface (11" by 10") and a deluxe sleek plastic coating, you'll be zooming across your screen in no time at all. Speed limit not enforced!

Our Price **$8.95**

Order Toll-free
800-MACDEV-1
(800-622-3381)

EXCLUSIVES!

FrameWorks Magazine

$8 per back-issue, subject to availability.

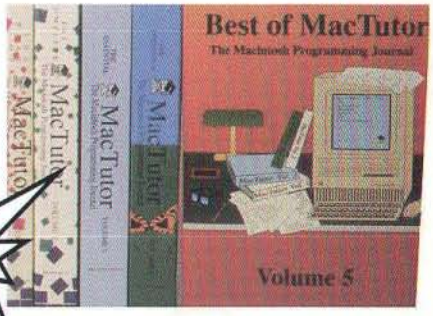
FrameWorks Source Code Disk

$10 per back issue, subject to availability.

MADACON '93 CD-ROM

The highlights of MADACON '93, including Mike Potel on Pink, Bedrock, MacApp, OODLs, and more. Slides, articles, demos, audio, and QuickTime. List $95.00 Our Price **$9.95** per set.

BLOW OUT!



Best of MacTutor

The Best of MacTutor Collection, Vols.3-5.
List $69.00 Our Price **$9.95** per set.

Developer DEPOT

Check out hundreds of more products on our Web site: http://www.devdepot.com

1-800-622-3381 • Outside U.S. 805-494-9797 • Fax: 805-494-9798

SEE RELATED PRODUCTS: Code Manager | SEE RELATED PRODUCTS: Inside Power Plant | SEE RELATED PRODUCTS: Inside CodeWarrior 8 | SEE RELATED PRODUCTS: C++ Programming with CodeWarrior | SEE RELATED PRODUCTS: PowerMac Programming Starter Kit | SEE RELATED PRODUCTS: Discover Programming For MacIntosh

CodeWarrior 9 Gold by Metrowerks

- Build applications for the Mac OS, Windows 95, Windows NT, Magic Cap, & the new Be OS.
- Includes C, C++ and Object Pascal compilers, source-level debuggers, object-oriented frameworks (PowerPlant, MacApp, MFC), Apple's MPW, complete online documentation and source code examples for all languages and platforms.
- The IDE software has been localized in eight languages plus English. CodeWarrior is sold on an annual subscription basis and has three major releases each year.
- **Includes a 1 year MacTech subscription.**

Our Price **$399.00**

SEE RELATED CATEGORY: Internet Related

SEE RELATED PRODUCTS: Learn C on the MacIntosh

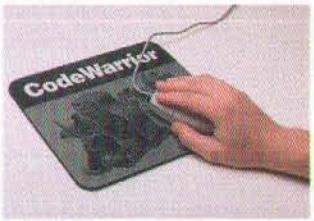
SEE RELATED PRODUCTS: Metrowerks CodeWarrior Programming

CodeWarrior 9 Bronze

- Build applications for the Mac OS on 68K machines.
- Includes C, C++ and Object Pascal compilers, source-level debuggers, object-oriented frameworks (PowerPlant, MacApp), Apple's MPW, complete online documentation and source code examples for all languages and platforms.
- The IDE software has been localized in eight languages plus English.
- Sold on an annual subscription basis and has three major releases each year.
- **Includes a 6 month MacTech subscription.**

Our Price **$149.00**

Lowest price guaranteed. Feel Good about your purchase! 30-day return policy.



CodeWarrior Mouse Pads

Fight your way through the night with these cool-looking pads!

- Arnold: Our Price **$8.95**
- Gears (not pictured): Our Price **$9.95**

CodeWarrior Wear (XL only)

You live it, you breath it... you might as well wear it!

- Black CodeWarrior Sweatshirt: Our Price **$29.95**
- "Blood, Sweat & Code" black long-sleeve shirt: Our Price **$7.95**
- Cross Platform white shirt (not pictured): Our Price **$14.95**
- Hawaii Five-0 shirt (not pictured): Our Price **$7.95**
- Arnold at work T-shirt (not pictured): Our Price **$9.95**
- Hat (black or white): Our Price **$14.95**
- Winter Hat (not pictured): Our Price **$14.95**

SmalltalkAgents® by Quasar Knowledge Systems

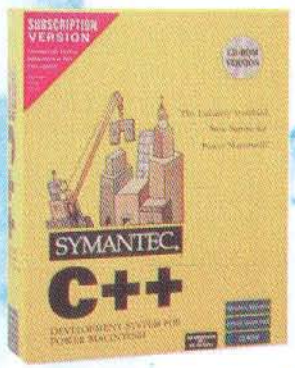
• Features a new generation of the Smalltalk language, QKS Smalltalk™. • Productivity is no longer measured in lines of code, but in project completion time. • Allows "live" direct manipulation of your objects. • Provides a dynamic, interactive and iterative development process. • Provides easy and full access to the features of the Mac OS™ and Mac Toolbox. • Link your non-Smalltalk code resources using our External Code Linking Toolkit™ (ECLT). • A sophisticated database for source code management provides an almost infinite variety of ways to cross-reference, access, view, and manipulate your code and objects. • Includes an Application Delivery Toolkit™ (ADT) that allows you to create royalty-free, stand-alone, double-clickable applications in just a matter of minutes. • Any component built in AO/S will function as an OpenDoc component or container and components from non-AO/S sources can be seamlessly integrated into the AO/S system.

Our Price **$695.00**

SEE RELATED CATEGORY: Internet Related

SEE RELATED PRODUCTS: Symantec C++ Programming
SEE RELATED PRODUCTS: Learn C++ on the MacIntosh
SEE RELATED PRODUCTS: Programming in Symantec C++
SEE RELATED PRODUCTS: Mastering theTHINK Class Library

Symantec C++ 8.4



- Native for Power Macintosh.
- Develop full-featured Power Macintosh applications quickly and easily!
- Environment includes: a true native Power Mac implementation of the C++ language, MrC/MrC++ compilers for fast Power Mac executable code, Visual Architect for fast, easy GUI generation, a new THINK project management system that supports complex applications, a new editor/browser , a powerful, easy-to-use source code debugger.
- Also includes: The industry-standard THINK Class Library!
- Next two updates free when you send in your registration card and more!

List $499.00 Our Price **$349.00**

Symantec C++ for 68k

- The standard in development languages. • Powerful combination of fully integrated visual tools and the latest in C and C++ compile technology. • Provides an object-oriented approach to application development. • Write code that is extensible, reliable, and maintainable. Includes: Integrated Environment with full source-code debugging, Incremental Linker which eliminates long link times THINK Class Library, AppleEvents, Visual Architect™, THINK Inspector, Project Models, Source-Code Control with integration with Apple's SourceServer (included), Support for Scripting, Powerful Standard Libraries which includes I/O Streams, ANSI standard C library, and sample programs. • Full source code is included – Create applications that run on 68K Macs and Power Macs (emulated These applications can easily be migrated to native Power Mac, by trading up to Symantec C++ for Power Mac.

List $369.00 Our Price **$99.00**

Think Pascal Version 4.0

by Symantec Corporation

- Great for professionals and students
- Fully integrated for rapid turnaround time – take advantage of System 7 capabilities
- Supports large projects, enhanced THINK Class Library, System 7 compatibility, superior code generation, and smart linking. • Includes four Macintosh disks, a user manual, and an object-oriented programming manual.

Our Price **$169.00**

LS Object Pascal CD-ROM by Fortner Research.

- Includes the world's first Object Pascal compiler for Power Macintos
- 100% compatible with Apple's MPW Pascal
- Combines the best of Apple's native development tools with innovative new technology developed at Language Systems
- Compiler options specify 68K or native PowerPC code generation
- CD includes: LS Object Pascal compiler, Universal Pascal Toolbox interfaces, fully loaded MPW 3.3.1, 68K and PowerPC source debug gers, PowerPC assembler, online documentation, Macintosh Tech Notes, and a special version of AppMaker by Bowers Development that generates native Pascal source code.

List $399.00 Our Price **$349.95**

MachTen – Power UNIX by Tenon Intersystems

- Dynamically linked shared libraries, memory mapped file access and integrated UNIX and Macintosh development tools.
- BSD 4.4 and conforms to the Federal Information Processing Standard 151-2 (the POSIX FIPS).
- Pre-emptive multitasking for UNIX applications and includes a full featured high-performance TCP/IP protocol stack that supports multi-homing and multi-casting, features not yet available even with Apple's new Open Transport.
- A complete UNIX software development environment with a source-level debugger and C, C++, and Fortran compilers all generating native PPC code.
- Also included is a high-performance X server and complete X11R5 X.

List $1,045.00 Our Price **$695.00**

SEE RELATED CATEGORY: Internet Related

Personal MachTen for 68K Macs

- MachTen UNIX for Macintosh (from Classic on up, including PowerBooks and Duos)
- A Mach-based Berkeley UNIX with pre-emptive multi-tasking
- Extends the Mac OS with UNIX networking and software development tools.
- Built-in internet services include domain name service, POP mail service, internet routing, SLIP & PPP, and Web service.

Our Price **$495.00** Also Available: MachTen XWindows Our Price **$350.00**

SEE RELATED CATEGORY: Internet Related

Mjølner BETA System

• Software development environment supporting object-oriented programming in the BETA programming language. • Uniquely expressive and orthogonal. • Unifies just about every abstraction mechanism – including class, procedure, function, coroutine, process and exception. • Includes: general block structure, strong typing, whole/part objects. The compiler: binary code generation, automatic garbage collection, separate compilation, interface to C, Pascal, and assembler. • The system: persistent objects, basic libraries with containers classes, platform-independent GUI application frameworks on Unix, Mac and Windows NT, metaprogramming system. • The Mjølner BETA System for Macintosh requires MPW (basic set) 3.2 or later. Package containing compiler, basic libraries, persistent store, GUI framework, and comprehensive documentation. (Other packages are also available).

List $295.00 Our Price **$249.99**

LS Fortran by Fortner Research

• 68K or PowerMac LS FORTRAN comes in two versions: one that generates 68K code (with or without FPU) and one that generates native code for the PowerMac • Source code is compatible between these two compilers • Programs originally developed with the 68K version can boost execution speed by 4 to 6 times by running the same program on a PowerMac • Supports all VAX intrinsic functions and data types, and virtually all VAX extensions • Built-in Debugging • Free Runtime Licenses • Full-featured editor, multiple window capabilities, and a powerful scripting language • Highly optimized, full ANSI standard, 32-bit clean, FORTRAN 77 compiler • Fully compatible with virtual memory under System 7 • Supports a variety of third-party tools to customize any FORTRAN environment • Power Tools include graphing software, subroutine libraries, and user interface tools • Free technical support.

Our Price **$695.00**

Fortran 77SDK by Absoft

For Power Macintosh includes a globally optimizing native compiler and linker, native Fx™ multi-language debugger, and Apple's MPW development environment.

• The compiler is a full ANSI/ISO Fortran 77 implementation, and includes all MIL-STD 1753 extensions, Cray/Sun-style POINTER, and several Fortran 90 enhancements • MRWE, Absoft's framework library is included in the MIG graphics library • Supports the native Macintosh PPC toolbox • Includes Absoft's Fx debugger which can debug intermixed FORTRAN 77, C, C++, and PPC assembler • The linker compiler, and debugger all run as native PPC tools, and produce Macintosh PPC executables

Our Price **$699.00**

LPA MacProlog

• Comprises a Edinburgh syntax Prolog compiler system set in an attractive multi-window development environment with an integrated program editor, graphical call-graph facilities and an interactive source-level debugger.
• Features high-level access to the Macintosh ToolBox for using graphics, dialogs, windows, icons, resources in a simple and versatile way.
• Includes interfaces to C and Pascal code resources.
• Enables the production of double-clickable distributable applications.
• Optional add-ons tools include flex, Prolog++, MacDBI for Oracle and the MacProlog Dialog Editor.

Programmer Edition **$745.00** Developer Edition (includes the run-time generator and distribution license) **$1500.00**

MacFortran II V 3.3 by Absoft

MacFortran II is a VAX/VMS compatible full ANSI/ISO Fortran 77 compiler including all MIL-STD 1753 extensions

• Comes with the latest version of MPW, and includes SourceBug, and SoftwareFPU
• Includes Absoft's Macintosh runtime Window Environment (MRWE) application framework, and MIG graphics library
• MacFortran II features improved 68040CPU support and is fully compatible with Power Macintosh under emulation

List $595.00 Our Price **$549.99**

BASIC for the Newton

• A fully interactive implementation of the BASIC programming language.
• Runs entirely on the Newton – no host is required.
• Create files, access the built-in soups, and the serial port for input and output.
• Work directly on the Newton, or through a connected Mac/PC and keyboard.

Our Price **$99.00**

SEE RELATED PRODUCTS:
Programming for the Newton using NewtonScript

SEE RELATED PRODUCTS:
Programming for the Newton using NS Basic

Order Toll-free
800-MACDEV-1
(800-622-3381)

Roaster DR2™

From Natural Intelligence Inc. Get the most out of Sun's new Java™ Programming Language!

SEE RELATED CATEGORY: **Dev. Environments**

- Write, test and run Java™ applets on the Macintosh in a full-featured Mac development environment
- Features include: project window that includes a finder-like view of packages, Macintosh native compiler, source code editor with powerful search features and intuitive use interface, runtime engine for quick and easy applet testing.
- Requirements : PowerPC, CD-ROM.

List $399.00 Our Price **$299.00**

TCP/IP Scripting Addition™

SEE RELATED CATEGORY: **Scripting**

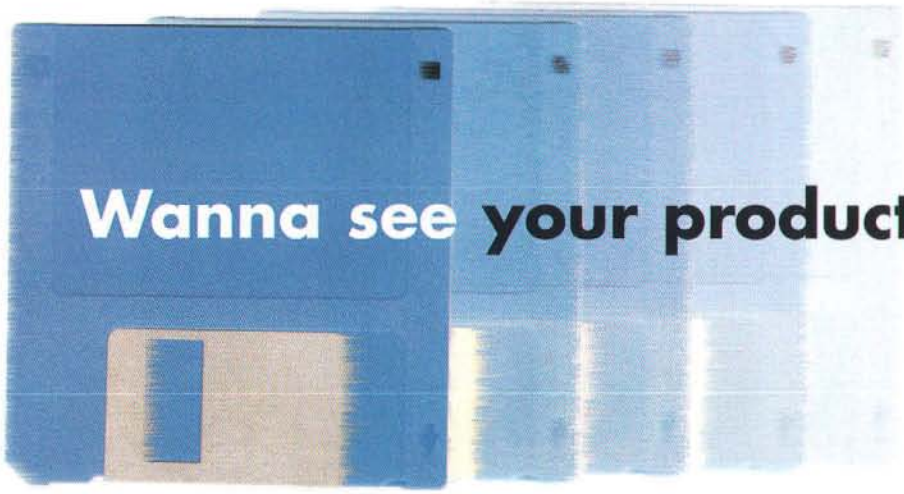
- Award-winning AppleScript scripting addition
- Allows you to write scripts using MacTCP™ commands in AppleScript™.
- Send e-mail or files through a script, check if users are logged on (via Finger), automate FTP, Gopher, NetNews, Telnet, and LPR, verify links in HTML documents, and quickly write many other TCP/IP client-server programs.
- Works with AppleScript 1.0 or later and MacTCP 2.0.4 or later. Compatible with Open Transport™ 1.1.

Our Price **$49.00**

SEE RELATED CATEGORY: **Tools, Libs & Utilities**

OOFILE HTML Writer

The first OODBMS framework to offer a complete solution for application authors. Replaceable backend database, currently Faircom's c-tree Plus for cross-platform royalty-free power. PowerPlant and other frameworks integrated with edit fields, database browsers and more. AppMaker users, generate complete applications. User-friendly syntax makes it easier to migrate from FoxPro and the non-OO world. Demo's on CodeWarrior and AppMaker CD's. **$900.00** for a once-off c-tree bundle, or **$1,095.00** 1year subscription. HTML and character-mode report-writer **$195.00**. Full GUI report-writer, including HTML, is **$495.00**. Mac Platform Bundle - includes all Mac frameworks, c-tree, and Report-Writer. **$1,495.00**



Get it in Developer Depot!

The fastest, cost effective way to get product off your shelves.

For information: • Voice: 805/494-9797 • Fax: 805/494-9798 • E-mail: marketing@devdepot.com

Developer DEPOT

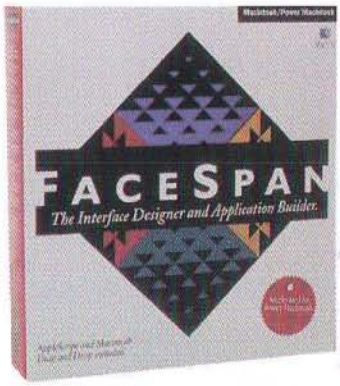
Can't find what you're looking for? Check our Web site: http://www.devdepot.com

1-800-622-3381 • Outside U.S. 805-494-9797 • Fax: 805-494-9798

DataScript

- The quickest, easiest and most cost-effective way to make your integrated AppleScript solutions database aware-today.
- Takes just six lines of AppleScript to make new or existing scripted solutions database aware, and fetch data from RDBM's such as Oracle, Sybase, DB2, or Informix.
- Easy to learn, easy to use, and easy to remember. "Inside DataScript" contains lots of easy to follow scripts to reuse in your own solution.
- You'll become productive very quickly, and once you're ready to ship, you'll find the licensing schemes very attractive.

List $249.00 Our Price **$229.99**



FaceSpan™ v2

Build applications quickly and easily!

- Extensible Rapid Application Designer (RAD).
- Combines an interactive, visual interface design environment with the object-oriented power of AppleScript or any OSA language.
- Integrate the capability of scriptable programs into your custom application.
- Display scrolling lists, popup menus, scrolling text, movies, multi-column tables, pictures, icons, buttons, and others.
- Program custom objects using Pascal or C.
- Includes a royalty-free distribution license, for unlimited runtime users, of your FaceSpan-based applications. Plus a FREE UPGRADE to the next version for registered users!

List $199.00 Our Price **$179.00**

Script Debugger by Late Night Software Ltd.

- A powerful and flexible AppleScript authoring tool – get the most from AppleScript!
- Advanced debugging environment offers single-step script execution with breakpoints.
- Script Debugger dictionary browser features a graphical view of objects provided by scriptable applications.
- Includes Late Night Software Scripting Additions – a collection of more than 70 new AppleScript commands, and Scheduler, a utility that allows you to launch scripts at pre-determined times.

List $129.00 Our Price **$119.99**

ScriptWizard™ 1.5

- Combines the power of a professional development environment with the Mac's ease of use.
- Compatible with all Apple® Open Scripting Architecture languages, including AppleScript™.
- Includes delivering testing and debugging facilities to improve your productivity.
- Emphasizes features that speed script development.
- Single-step scripts with true statement-level stepping, watch variable values as scripts execute, jump instantly to frequently used places in a script and find and replace specific text.

List $89.00 Our Price **$84.95**

Scripter®

- Powerful and easy to use, for both novices and experts.
- Includes Superior vocabulary access – point-and-click assembly of commands and object specifications; command window for experimentation.
- Includes Shortcuts and extended editing capabilities – extensive drag-and-drop, six-function find-and-replace; navigation markers; script library collection facility; many other timesavers for faster scripting.
- Interactive debugging – comprehensive variable watcher, expression evaluation, enhanced trace log, and real single step debugging!

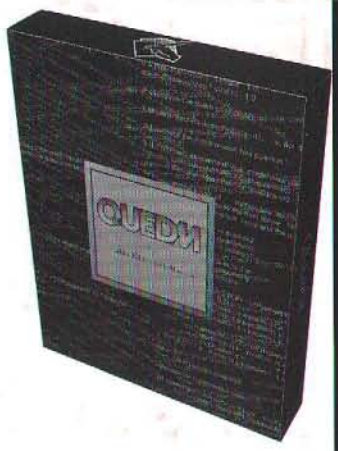
List $199.00 Our Price **$179.00**

SEE RELATED CATEGORY:
Internet Related

BBEdit 3.5.2 from Bare Bones Software

- A powerful, easy-to-learn text editor.
- Adds new features for HTML coders, including a spelling checker and HTML tag palette.
- Accelerated for Power Macintosh; dragging supported everywhere; Internet Config aware; PowerTalk aware.
- Integrated support for Symantec's IDE, Metrowerks CodeWarrior, THINK Reference 2.x, MPW ToolServer, and most other environments.
- Many UNIX style tools, including "grep" searches, file comparisons, and sorting. Multi-file search and replace.
- PopUpFuncs feature lets you jump to a function from a menu.

List $119.00 Our price **$94.99**



QUED/M 3.0 by Nisus Software

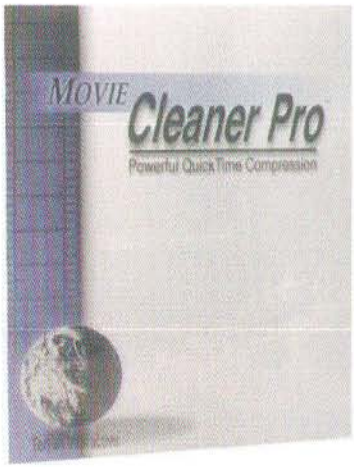
- The programmer's text editor that defined the industry standard for speed and efficiency.
- Power PC native.
- Features integrated support for Symantec C/C++, Metrowerks CodeWarrior 6, and MPW.
- Supports all the major development environments on the Macintosh.
- Dozens of powerful editing features, including unlimited undo and redo, macro language, scripting, text folding, ten editable/appendable clipboards, markers, displaying text as ASCII codes, dynamic coloring of C/C++ keywords/comments, rectangular and non-contiguous selection.
- Includes Celestin Company's APPRENTICE 4.

List $149.00 Our Price **$89.00**

CMaster 2.0 by Jersey Scientific

- Installs into THINK C 5 / 6 / 7 and Symantec C++ for Macintosh and enhances the editor.
- Use its function popup to select a function and CMaster takes you right to it!
- Multiple clipboards and markers, a Function Prototyper, and a GoBack Menu which can take you back to previous editing contexts.
- Almost all features bindable to the keyboard, along over a hundred keyboard-only features like "Add New Automatic Variable." Glossaries, AppleScript and ToolServer support, Macros, and External Tools you create too!

Our Price **$129.95**



SEE RELATED PRODUCTS:
Quick Time Starter Kit

Movie Cleaner Pro by Terran Interactive

SEE RELATED PRODUCTS:
Quick Time Official Guide

- Compress QuickTime movies
- Powerful and easy-to-use
- Includes: drag and drop batch processing, suspend and resume, high quality crop and resize, A/V fades, and much more.
- Automatically suggests the best settings for your application.
- Great for both beginners and experts
- Essential for CD-ROMs or Web sites

Our Price **$129.95**

One & 2-Day SHIPPING (call for details)

Developer DEPOT

Check out hundreds of more products on our Web site: http://www.devdepot.com

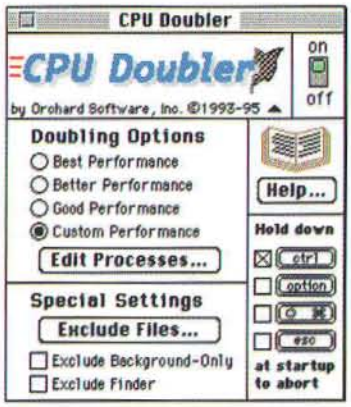
1-800-622-3381 • Outside U.S. 805-494-9797 • Fax: 805-494-9798

CodeManager by Metrowerks

SEE RELATED CATEGORY:
Dev. Environments

- Source code control system for the Macintosh.
- Based on and compatible with Microsoft Visual SourceSafe version 4.0.
- Works with any type of binary file including graphic, database, library and executable files.
- Includes support for multi-platform projects, security features, reverse delta versioning of files, comment areas for each revision, project analysis and reporting tools.
- Sold on a subscription basis with two future product updates.
- **Includes a 1 year MacTech subscription.**

Our Price **$399.00**



CPU Doubler by Orchard Software

SEE RELATED PRODUCTS:
Cron Manager

- Performance enhancement utility for the Macintosh.
- Increases the speed of your computer by 100%.
- Works on both the PowerPC and 68K Macintosh.
- Manages computer throughput using a proprietary scheduling algorithm.
- Ensure optimal performance and compatibility.

Our Price **$79.95**

CLImate by Orchard Software

- Command line interface that lets you communicate with your Macintosh using English commands to create, delete, rename, and move files and folders.
- Start applications, format disks, restart your computer, etc.
- Supplements the Finder.
- Includes a BASIC interpreter that lets you script your Macintosh without AppleScript.
- Includes advanced programming constructs: repeat loops, if/then/else conditionals, subroutine calls, etc.
- Implements wildcard characters, enabling you to work on groups of files.
- Comes bundled with sample programs and full documentation.
- Includes a free copy of Cron Manager – chronological event management utility.

Our price **$59.95**

File Genie Pro by Duet Development

FILE genie PRO

- File Genie Pro lets you search ALL your developer CD-ROMs at once!
- Every Dev CD, OS SDK, CodeWarrior CD, E.T.O., and Bookmark CD you insert is cataloged automatically. A typical Dev CD with 10,000 files is cataloged in under 10 seconds.
- You can also catalog opticals, SyQuests, and floppies.
- File Genie Pro quickly finds any file on any disk by searching hard disks, network servers, and catalogs. Open, show, print, and more. View text and graphics files without opening other applications.
- When acting on an ejected disk file, File Genie Pro tells you which disk to insert, then continues automatically.

Our Price **$89.00**

Order Toll-free
800-MACDEV-1
(800-622-3381)

Guide Composer™ by StepUp Software

SEE RELATED PRODUCTS: Real World AppleGuide

SEE RELATED PRODUCTS: AppleGuide Complete

SEE RELATED PRODUCTS: Danny Goodman's AppleGuide Starter Kit

- Create powerful Apple Guide help systems for any new or existing Macintosh application.
- Great for commercial developers, shareware developers, in-house developers, and consultants.
- Provides a WYSIWYG development environment: Guide content is developed in Guide windows.
- Design topics, phrases, and panels in the same format as the user will use them. Features are WYSIWYG interface, Topics, phrases, and hierarchical phrases, Coach marks, Fully-Integrated with Apple's Guide Maker (distributed with Guide Composer), compiles scripts automatically, PICTs in Panels, Generated Guide scripts are modifiable.
- Compiled files are 100% AppleGuide-compatible and royalty-free. Easy-to-use.

Our Price **$99.00**

STEPUP SOFTWARE

ScriptGen Pro by StepUp Software

- Installer script generator which requires no programming or knowledge of Rez.
- Supports StepUp's InstallerPack, StuffIt compression, custom packages, splash screens, network installs, Rez code output, importing resources, and AppleEvent link w/MPW

Our price **$169.00**

Step-Up Installer Pack by StepUp Software

- Package of several Installer "atoms" that let developers incorporate graphics, sounds, file compression and custom folder icons into installation scripts.
- Compression formats supported are Compact Pro & Diamond.
- Each atom also available separately.
- Compression requires additional licensing.

Our price **$219.00**

SEE RELATED PRODUCTS: CPU Doubler

CronManager
by Orchard Software

- Implements the UNIX Cron facility.
- Open any Macintosh file on a given date and time.
- Simple interface.
- Works with any Macintosh file.
- Cron Manager bundled with CLImate,

Our price **$26.95**

Tenon Ported Application CD

- Contains hundreds of applications that have been pre-compiled to run on Personal or Professional MachTen 68k systems, such as Unix, Perl, and TCL/TK
- Includes many internet servers such as Gopher, WAIS Ported Application CD Vol. 1 (for 68K Macs).

Our price **$49.95**

SEE RELATED CATEGORY: Scripting

Rosanne

- A collection of utilities which offer the user complete control over raw data.
- Sort files, extract selected records, summarize frequency counts, create sample files, perform matching on multiple files, and reformat data to new specifications, all on the desktop, and even on files of a million records or more.
- Supports AppleScript™, enabling the user to link several actions together to complete an entire process.
- Recordable – users may perform a series of actions, and see them translated directly into AppleScript commands.
- Supports multi-tasking and background processing.
- Rosanne Utilities: Copy, Format, Select, Recode, Sort, Match, and Aggregate

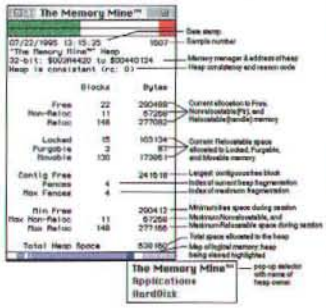
Our Price **$595.00**

SEE RELATED CATEGORY: Dev. Environments

Memory Mine by Adianta

- Monitor heaps, identify problems such as memory leaks, and stress test applications
- Active status of memory in a heap is sampled on the fly: allocation in non-relocatable (Ptr), relocatable (Handle) and free space is shown, as are heap corruption, fragmentation, and more
- Allocate, Purge, Compact, and Zap memory lets users stress test all or part of a program
- Works on Macintoshes with 68020 or later and System 7.0 or later.

List $99.00 Our Price **$94.99**



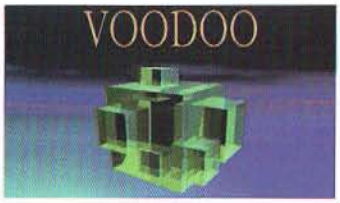
QC by Onyx Technology. High performance runtime stress testing for applications.

- Tests include heap checks, purges, scrambles, handle/pointer validation, dispose/release checks, write to zero, de-reference zero as well as other tests like free memory invalidation and block bounds checking.
- Extremely user friendly – ideal for non-programmer testers.
- Also available in Japanese.

List $99.00 Our price **$94.99**

SEE RELATED CATEGORY:
Dev. Environments

VOODOO



- Version control tool for the simple and clear management of projects in which files are created in numerous versions (variants and revisions).
- Allows both variant and revision control, and it manages not only variants and revisions of single files, but of a whole software project (multi files, multi users, multi variants, access rights, etc.)
- Graphical user interface and is not only suitable for mere source code control but can handle all different kinds of files with amazing compression rates: typical size of delta between arbitrary files 5%
- Please note special prices for multiple copies:

Single license **$229.00**; 2 pack **$359.00**; 5 pack **$799.00**; 10 pack **$1369.00**; 20 pack **$2399.00**
Additional pricing available on request.

Last Resort Programmers Edition

- Records every keystroke, command key and mouse event (in local coordinates) to a file on your hard disk
- Great for program testing & debugging, and for technical support and help desks
- Last Resort keystroke files and recovers what you typed – great for data losses caused by power failure, crashes, or accidental deleting.

Our price **$74.95**

Spyer by InCider

- Easy to use tool that records all actions (including mouse movement) you perform on a Macintosh computer and then replays them at your preferred speed.
- Recorded data can be saved in files for future use.
- Works as a background process with any Macintosh application and is triggered by user defined Hot Keys.
- Enables the "Continuous Redo" utility and is especially useful for software testing and demonstration.

Our price **$39.00**

LJ Profiler by Lars Jordebo Datakonsult

- Supports profiling of C++ 68K and PowerPC applications compiled with CodeWarrior, CFront or Symantec C++.
- Based on active profiling, i.e. profiling code called at function enter and exit, the browser application lets you follow call chain timings in hierarchical views or separate windows.
- Collect, organize, compare and save profiling data from different versions of your application into a project. Scriptable and recordable with full access to most internal data structures.
- Optional remote profiling and tracking of segment and stack usage. Full source code to what you link into your application.

Our price **$295.00**

PowerTap™

- Accelerates software by tapping into multiple processors.
- Easy API – submit tasks and retrieve results!
- Full error recovery system plus scheduling algorithms for optimal assignments and fastest possible execution.
- Compatible with all Macintosh hardware, software and major compilers.

PowerTap/2, 2 remotes edition List $1,200.00 Our Price **$1,099.99**
PowerTap/5, 5 remotes edition List $1,900.00 Our Price **$1,799.99**
PowerTap/n, unlimited edition List $2,700.00 Our Price **$2,499.99**

One & 2-Day SHIPPING (call for details)

StoneTable

- Replaces all functions found in list manager
- Variable size columns/rows; different font, size, style, forecolor, backcolor per cell; sort, resize, move, copy, hide columns/rows; edit cells/titles in place; titles for columns/rows; multiple lines per cell; grid line pattern/color; greater than 32k data per table; up to 32k text per cell; support for balloon help and binary cell data. Versions for Think C, Think Pascal, MPW C, MPW Pascal, CodeWarrior 6 C.

First compiler **$150.00**, additional compilers **$50.00**

StoneTable Extra

- Drag selected cells within table or to other tables.
- Add rows as part of drag; popup menus or check boxes in cells; variable width grid lines; move/drag/resize table in window; clipboard operations on multiple cells.
- Requires StoneTable.

First compiler **$50.00**, additional compilers **$25.00**

StoneTable and StoneTable-Extra for PowerPC

- An upgrade for owners of StoneTable
- Same functionality as 68K libraries.
- Versions for MPW C and CodeWarrior 6 C.
- Must have 68K libraries.

StoneTable **$100.00** StoneTable-Extra **$25.00**

PictureCDEF 1.3 by Paradigm Software

- Professional-level CDEF for creating custom graphical buttons (8-64 pixels) – used in products by Adobe, ProVue, STF Technologies and others!
- Multi-monitor and bit-depth sensitive.
- The button graphic (cicn, ResEdit) can be changed at runtime and even animated with a call-back routine.
- Create distinct buttons in seven variations: MultiState, PushButton, FlexiButton, ToggleButton, ChkButton, PushPictButton and TogglePictButton.
- Manual, sample code and MacApp 3.0 support included.

Full source code: **$95.00** Object code: **$45.00**

SpellsWell 7 1.0.4

- Award-winning, comprehensive, practical spelling checker that works in batch mode or within applications that incorporate the Apple Events Word Services protocol (e.g., Eudora, WordPerfect, Communicate!, and Fair Witness).
- Checks for spelling errors as well as common typos like capitalization errors, spaces before punctuation, double word errors, abbreviation errors, a/an before vowel/consonant, etc.
- MacTech orders include developer kit with Writeswell Jr., a sample AppleEvents Word Services word-processor and its source code.

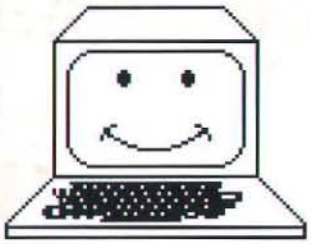
Our price **$74.95**

MacWireFrame by Amplified Intelligence

- Create your own virtual reality application with MacWireFrame, a virtual reality application framework.
- Includes a complete library of object oriented graphics routines.
- Easy to understand application frame work, plus an example application program that lets you start solid modeling right away.
- Complete with fully documented source code.
- Guaranteed $49.99 upgrade to the soon to be released, scriptable, MacWireFrame 5.0.

List $299.00 Our Price **$75.00**

Order Toll-free
800-MACDEV-1
(800-622-3381)



B-Tree HELPER™ 2.2

- Inexpensive database engine for Macintosh programmers in C source code.
- Uses contiguous fixed length blocks.
- Expands the file as necessary and contracts files when possible.
- Inserts and deletes keys in one or more B-Trees.
- Finds keys equal to, less than, or greater than a given value in a few hundredths of a second.
- Finds lists of records whose keys are equal to, less than, or greater than a given value or are in a range of values.

Our Price **$150.00**

Q3S/3dPane/SmartPane

ViviStar CONSULTING

- Full featured 3d graphics library.
- Polyhedra; Gouraud shading; stereoscopic projections; pipeline access; animation and model interaction support; a "triad mouse" to map 2d mouse movement to 3d; and all the regular 3D features.
- 3dPane provides integration with the TCL and provides a view orientation controller. SmartPane provides TCL offscreen image buffering, flicker free animation, and QuickTime movie recording. SmartPane functions in 3d or 2d scenarios.
- All work with C++ compilers or ThinkC 6 and compile to PowerPC or 68K target machines.

Our Price **$192.00**

dtF

- True relational database system for Apple Macintosh computers.
- Provides a powerful choice for developers who want to create database centered applications with no performance trade-offs.
- Features SQL, full transaction control, error recovery, single user, client server architecture and multi-platform support including DOS, Windows, OS/2 and UNIX.
- The C/C++ API is identical and fully portable across all supported platforms.
- Third-party vendors supporting dtF will be able to offer a variety of advanced features and benefits to their customers royalty free.
- Tools are included for importing, exporting, creating and managing databases and users.
- Supported development environments include: Symantec, MPW, Metrowerks and more. Mac/SDK

List $695.00 Our Price **$679.99**

OOFILE

SEE RELATED PRODUCTS:
OOFILE HTML Writer

- First OODBMS framework to offer a complete solution for application authors.
- Replaceable backend database, currently Faircom's c-tree Plus for cross-platform royalty-free power.
- PowerPlant and other frameworks integrated with edit fields, database browsers & more. AppMaker users, generate complete applications.
- User-friendly syntax makes it easier to migrate from FoxPro and the non-OO world.
- **$900.00** for a once-off c-tree bundle, or **$1,095.00** 1-year subscription.
- HTML and character-mode report-writer. Our Price **$195.00**.
- Full GUI report-writer, including HTML. Our Price **$495.00**.
- Mac Platform Bundle - includes all Mac frameworks, c-tree, and Report-Writer. Our Price **$1,495.00**

NeoAccess™

- Full-featured object database engine for use in Macintosh, Windows, Unix and DOS based C++ applications.
- Extended binary trees and binary search algorithms tuned for short access times; dynamically combined, collapsed, and compressed indices; object caching for instant access to previously used objects.
- Build fast, powerful applications in record time!

Our price **$749.00**

Lowest price guaranteed. Feel Good about your purchase! 30-day return policy.

AppMaker

- Develop the user interface for a Macintosh application using the original interface builder.
- Just point and click to design your application.
- Creates resources and generates excellent source code.
- Supports most development environments including Metrowerks, Symantec, or MPW; C, C++, or Pascal; procedural or object-oriented, using PowerPlant, TCL, or MacApp.
- The generated code uses the Universal Headers to provide PowerMac compatibility.
- Great tool for beginners to learn object-oriented and Macintosh Toolbox programming techniques.
- Includes one-year subscription on CD.

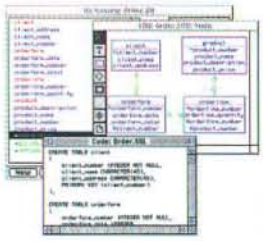
List $299.00 Our Price **$279.99**

MacDesigner/Expert by Excel Software

- Supports software engineering methods with the capabilities of MacDesigner plus multi-task design.
- Integrated requirement database provides traceability from requirement statements to design diagrams, code or test procedures.
- Suited to design or maintenance of real-time, multi-tasking software projects.

Expert Demo **$79.00**; Expert Product **$1595.00**; MacDesigner Demo **$79.00**; MacDesigner **$995.00**

MacAnalyst/Expert by Excel Software



- Supports software engineering methods with the capabilities of MacAnalyst plus state transition diagrams, state transition tables, decision tables and process activation tables
- An integrated requirement database provides traceability from requirement statements to analysis or design diagrams, code or test procedures
- Well suited to the analysis and design of real-time or requirements driven projects.

Expert Demo **$79.00**; Expert MacAnalyst **$1595.00**; MacAnalyst Demo **$79.00**; MacAnalyst **$995.00**

MacA&D by Excel Software

- Combines the capabilities of MacAnalyst/Expert and MacDesigner/Expert into a single application.
- Supports structured analysis and design, object-oriented analysis and design, real-time extensions, task design, data modeling, screen prototyping, code editing and browsing, reengineering, requirement traceability, and a global data dictionary.

MacA&D Demo **$149.00**; MacA&D **$2995.00**

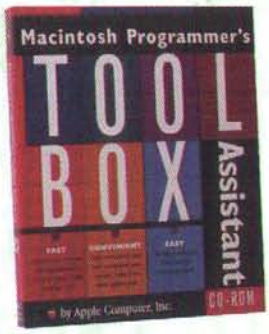
ICONIX Power Tools - Ten modules that can be used either as stand-alones or combined to form a custom, integrated development environment with dictionaries and Language Sensitive Editors

- **ObjectModeler** supports several object-oriented methods with four graphical editors
- **FastTask** provides real-time extensions in the form of State Transition Diagrams and their equivalent matrix representation
- **DataModeler** supports object-oriented analysis and includes an Entity Relationship Attribute (ERA) Diagram Editor
- **FreeFlow** provides support for DeMarco Structured Analysis with real-time extensions and several object-oriented methods.
- **PowerPDL** supports detailed algorithm design using pseudocode.
- **AdaFlow** supports Ada-oriented design and Ada development
- **QuickChart** supports Structure Chart Editing
- **SmartChart** supports Detailed Design with automatic Structure Chart Generation from Program Design Language, Processes Hierarchy Charts with FreeFlow
- **ASCII Bridge** enables PowerTools repository information to be merged, imported, and exported in ASCII or CDIF format and provides an Interleaf interface.
- **CoCoPro** implements Boehm's Constructive Cost Model (CoCoMo) technique for estimating costs of software projects.

PowerTools is available both as individual modules or in PowerPack bundles of 6, 8 or 10 distinct modules of your choice: Each individual module is **$1,495.00**.
PowerPack/6: **$5,995**; PowerPack/8: **$6,995.00**.
PowerPack/10: **$7,995.00** (Full Product Line).
Upgrade Service: 15% of purchase price, good for one calendar year upgrade.
Training and Consulting: **$3,000.00** per day, plus expenses.

Inside Macintosh®: Programmer's Toolbox Assistant CD-ROM

Instant electronic access to Inside Macintosh essentials.



- Get quick access to reference pages for over 4,000 Toolbox calls in your system software from their development environment.
- Essential information for Macintosh software developers.
- Hypertext links allow programmers to view related topics easily.
- The ultimate electronic reference tool for Macintosh programmers.

List $99.95 Our Price **$89.95**

Inside Macintosh®: Overview by Apple Computer, Inc.

- An overview of Macintosh programming fundamentals and a road map to the New Inside Macintosh library.
- Covers various programming tools and languages, compatibility guidelines and considerations for worldwide development. 176 pages.
- Great for beginners!

List $22.95 Our Price **$20.65**

Inside Macintosh®: Macintosh Toolbox Essentials

by Apple Computer, Inc.

- Covers the heart of the Macintosh. The toolbox enables programmers to create applications consistent with the Macintosh "look and feel".
- Describes Toolbox routines and shows how to implement essential user interface elements, such as menus, windows, scroll bars, icons and dialog boxes. 880 pages.

List $34.95 Our Price **$31.45**

Inside Macintosh®: More Macintosh Toolbox by Apple Computer, Inc.

- Managers discussed include Help, List, Resource, Scrap and Sound.
- Covers other Macintosh features such as how to support copy and paste, provide Balloon Help, play and record sound and create control panels.

List $34.65 Our Price **$31.45**

Inside Macintosh®: Files by Apple Computer, Inc.

- Describes the parts of the operating system that allow you to manage files.
- Explains how your application can handle the commands typically found in a File menu.
- Provides a reference to the File and Alias Managers, the Disk Initialization and Standard File Packages. 510 pgs.

List $29.95 Our Price **$26.95**

Inside Macintosh®: Operating System Utilities by Apple Computer, Inc.

- Describes parts of the Macintosh Operating System that allow you to manage various low-level aspects of the operating system.
- Shows how to get information about the operating system, manage operating system queues, handle dates and times, control the settings of the parameter RAM, manipulate the trap dispatch table, and receive and respond to low-level system errors.

List $26.05 Our Price **$23.45**

Order Toll-free
800-MACDEV-1
(800-622-3381)

Inside Macintosh®: Processes by Apple Computer, Inc.

- Describes the parts of the Macintosh operating system that allow you to control the execution of processes and interrupt tasks.
- Shows how you can use the Process Manager to get information about processes loaded in memory.
- A reference for the Vertical Retrace, Time, Notification, Deferred Task, and Shutdown Managers. 208 pages.

List $22.95 Our Price **$20.65**

Inside Macintosh®: Memory by Apple Computer, Inc.

- Describes the parts of the Macintosh operating system that allow you to manage memory.
- Provides detailed strategies for allocating and releasing memory, avoiding low-memory situations, reference to the Memory Manager, the Virtual Memory Manager, and memory-related utilities. 296 pages.

List $24.95 Our Price **$22.45**

E-Mail Essentials by Ed Tittel & Margaret Robbins

- A hands-on guide to the basics of e-mail.
- Covers everything from the installation of e-mail to the maintenance and management of e-mail hubs and message servers.
- For the casual e-mailer and the networking professional. 250 pp.

List $24.95 Our Price **$22.45**

The Computer Privacy Handbook

- A practical guide to e-mail encryption, data protection, and PGP privacy software.
- Learn the practical steps you can take to safeguard your electronic security.

List $24.95 Our Price **$22.45**

Cyberpunk Handbook, The Real Cyberpunk Fakebook

by St. Jude, R.U. Sirius, and Bart Nagel. Published by Random House.

Are you or someone you know a Cyberpunk? Read this book and you'll know!

List $9.95 Our Price **$8.95**

The Elements of E-Mail Style

by Brent Heslop and David Angell

- The rules of the road in the e-mail age.
- How to write solid, effective E-Mail and avoid common pitfalls. 208 pages.

List $14.95 Our Price **$13.45**

The Instant Internet Guide

- An Internet jump-start – how to access, use and navigate global networks, 224 pages.

List $14.95 **$13.45**

SEE RELATED CATEGORY: **Internet Related**

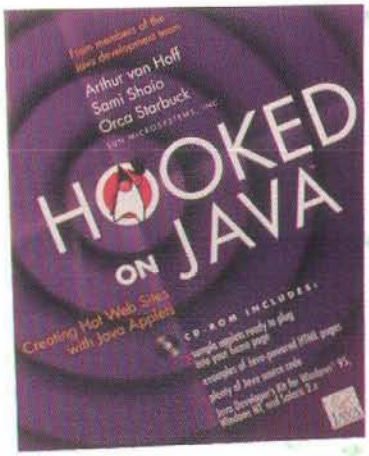
Planning and Managing Websites

- The definitive guide to setting up and running a Web site on the Macintosh.
- Learn everything you need to know about using WebSTAR, the best known HTTP server software and its shareware predecessor MacHTTP.
- Write CGI applications for your server – in AppleScript and in C.
- CD includes a special version of WebSTAR, plus tons of useful software.

List $39.95 Our Price **35.96**

Includes CD-ROM!

SEE RELATED CATEGORY: **Internet Related**



Hooked on Java

- Written by the Java development team at Sun.
- An introduction to using applets, for Web administrators, designers, and developers.
- Demonstrates how to use applets in your own pages.
- Includes a concise introduction to the Java language, and a CD with tools.

List $29.95 Our Price **$26.95**

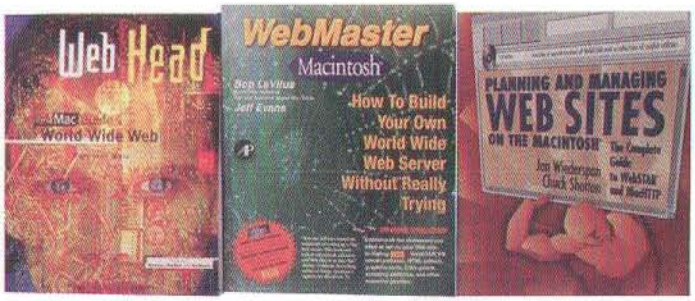
Lowest price guaranteed. Feel Good about your purchase! 30-day return policy.

Web Weaving

- Provides the guidance to create and maintain web sites that are easily navigated, scaleable, and maintainable.
- Covers the three major platforms for Web pages – UNIX, Windows, and the Mac – shows how to use software tools and utilities that are important in setting up Web sites.
- Learn how to install and configure Web servers, use authoring tools and converters, implement security schemes, and integrate multimedia features such as sound, video, and graphics into Web pages.
- Learn to organize and present the information in Web sites. Includes planning for growth, building in maintenance schemes, catering to users' needs, and creating a navigable, logical underlying infrastructure.

List $24.95 Our Price **$22.45**

SEE RELATED CATEGORY: **Internet Related**



Webmaster Macintosh

SEE RELATED CATEGORY: **Internet Related**

- A fast and easy guide for turning your Macintosh into a World Wide Web site – in a matter of hours!
- Includes CD!

List $29.95 Our Price **$26.95**

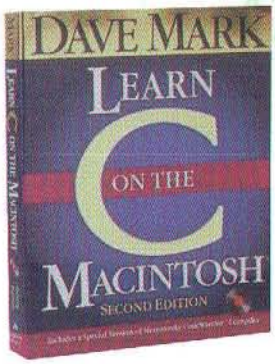
Includes CD-ROM!

WebHead Mac Guide

SEE RELATED CATEGORY: **Internet Related**

- Get the most from the Web.
- Easy-to-understand, great for beginners, and online veterans.
- Includes instructions on how to build your own home page, posting pages on the Web, and avoiding common HTML mistakes.

List $24.95 Our Price **$22.45**



Learn C on The Macintosh Second Edition By Dave Mark

- New revised edition.
- Easy-to-understand – everything you need to start programming!
- Updated and enhanced exercises that lead you step by step. You'll learn function, variables, pointers datatypes, data structures, file input and output and more!
- Includes CD-ROM with Metrowerks CodeWarrior™ Lite – the hottest Macintosh programming environment (including a PowerPC version).

List $34.95 Our Price **$31.45**

SEE RELATED CATEGORY: **Dev. Environments**

Discover Programming for Macintosh

- Includes full working version of CodeWarrior along with three online tutorial books and Dave Mark's "Learn C on the Macintosh" converted to AppleGuides.
- Includes C, C++ and Object Pascal compilers for generating 68K Macintosh code, source-level debuggers, object-oriented frameworks (PowerPlant, MacApp), Apple's MPW, complete online documentation and source code examples for all languages and platforms.
- The IDE software has been localized in eight languages plus English. This product is not sold as a subscription.
- **Includes a 3 month subscription to MacTech Magazine.**

Our Price **$79.00**

SEE RELATED CATEGORY: **Dev. Environments**

Programming Primer Macintosh Vol 1 by John Whittle and Judy May

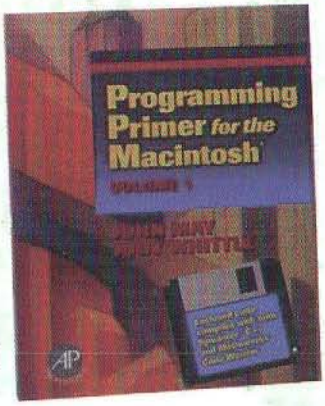
- An introduction to Macintosh programming, using C++ as the example language
- Provides realistic, easy to follow, programming examples designed to work with either Symantec® C++ or Metrowerks® CodeWarrior™
- Includes one 3.5" disk with source code for the programming examples, plus numerous public domain utilities to use with each compiler

List $37.95 Our Price **$34.15**

One & 2-Day SHIPPING (call for details)

Macintosh C Programming Primer Volume I

Second Edition, Inside the Toolbox Using THINK C by Dave Mark and Cartwright Reed



- Updated new edition of the Macintosh programming best seller.
- System 7, new versions of THINK C and ResEdit.
- Learn how to use the resources, Macintosh Toolbox and interface to create stand-alone applications.
- 672 pages.

List $26.95 Our Price **$24.25**

Macintosh C Programming Primer Volume II

Mastering the Toolbox Using THINK C by Dave Mark.

- Covers advanced topics such as: Color QuickDraw, THINK Class Library, TextEdit, and the Memory Manager: 528 pgs.

List $26.95 Our Price **$24.25**

Learn C++ on the Macintosh by Dave Mark.

SEE RELATED CATEGORY:
Dev. Environments

- Basic syntax of C++ and object programming.
- Learn how to write, edit, and compile your first C++ programs.
- Features key C++ concepts such as derived classes, operator overloading, iostream functions and more.
- Includes a special version of Symantec C++ for Macintosh. Book/disk package with 3.5" 800K Macintosh disk. 400 pages.

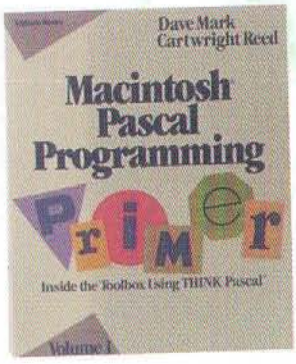
List $36.95 Our Price **$33.26**

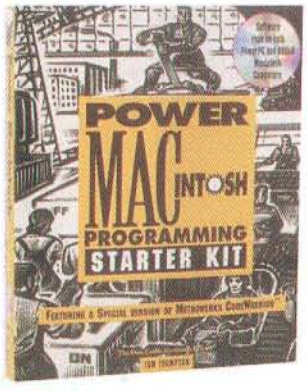
Macintosh Pascal Programming Primer Volume I

Inside the Toolbox Using THINK Pascal by Dave Mark and Cartwright Reed.

This tutorial shows programmers new to the Macintosh how to use the Toolbox, resources, and the Macintosh interface to create stand-alone applications with Symantec's THINK Pascal. 544 pages.

List $26.95 Our Price **$24.25**





Power Macintosh Programming Starter Kit

by Tom Thompson.

SEE RELATED CATEGORY:
Dev. Environments

- Enter the world of the PowerPC chips.
- Get the scoop on the microprocessors, the RISC architecture, and how to write native code and emulation operations to create software for the Macintosh PowerPC.
- CD-ROM includes a unique compiler for writing code easily.

Includes CD-ROM!

List $39.95 Our Price **$35.10**

Macintosh Programming Secrets 2nd edition

By Scott Knaster, and Keith Rollin

- Macintosh Programming Secrets is divided in two parts.

Part 1, "Concepts and Ideas", discusses the evolution of the Macintosh and the standards, customs, and software that shape the system as well as the Macintosh user interface.

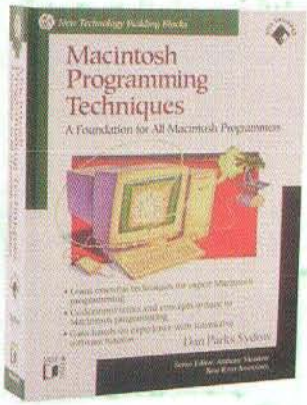
Part 2 "Technical Adventures", presents the skeleton of an application, and then builds upon that framework to describe how to: • Create fancy dialogue boxes • Utilize the new 32 bit QuickDraw developments • Track the mouse with "marching ants" • Manage multiple windows with the Window manager • Copy files within a program • Install the worlds strangest spinning cursor.

List $31.95 Our Price **$28.76**

Order Toll-free
800-MACDEV-1
(800-622-3381)

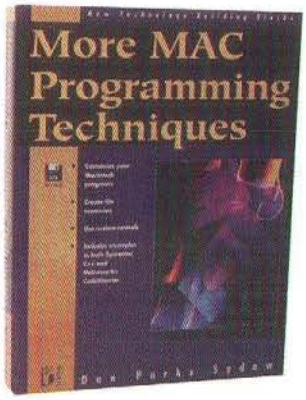
Macintosh Programming Techniques

by Dan Sydow (Series Editor: Tony Meadow)



- For Macintosh programmers as well as those making the transition from DOS, Windows, VAX or UNIX.
- Emphasizes programming techniques for better code – regardless of language!
- Guides you through Macintosh memory management, QuickDraw, events and more, using sample program in C++.
- Includes disk with an interactive tutorial, plus reusable C++ code.

List $34.95 Our Price **$31.95**



More Mac Programming Techniques

- Hands-on guide and tutorial, expands on the basic foundation of programming to develop truly powerful applications.
- Take a detailed look at the units of a Macintosh program – from the INITs to custom controls.
- Packed with solid tricks and techniques.
- Learn file resources from the ground up, and how to build custom menus, add custom controls, including buttons, and slider controls, MDEF, CDEF resources and much more.

List $39.95 Our Price **$35.95**

Programming with AppleTalk by Michael Pierce

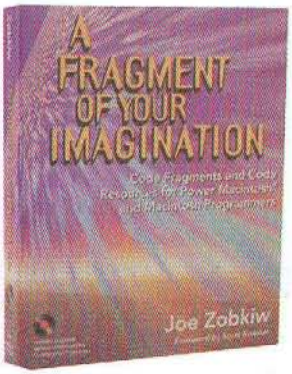
- Programming with AppleTalk is the hands-on guide to understanding and working with AppleTalk.

Topics covered include:

- How to create applications and system extensions that run with AppleTalk.
- AppleTalk protocols and the protocol stack, transport media, the Preferred AppleTalk Intefrace, and the storage management.
- Numerous working code examples walk you through using RDEV, INIT, NBP, ATP, and ADSP You will also learn the use of: Synchronous, and asynchronous calls, How to avoid heap fragmentation, And how to configure a Chooser Interface.

List $45.00 Our Price **$26.00**

A Fragment of Your Imagination by Joe Zobkiw



- Packed with useful code fragments for the Macintosh and Power Macintosh.
- Hard to find information about techniques used to structure and build fat, safe fat, and accelerated code resources.
- All code is reusable and is provided on the disc, along with Metrowerks Code Warrior Lite. Book/CD-ROM, 528 pages.

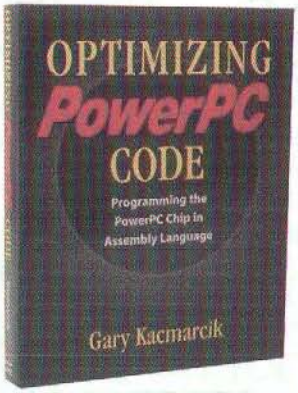
List $39.95 Our Price **$35.96**

Programming QuickDraw

- Learn to build color pictures on the Mac
- Learn to maintain the highest degree of compatibility for you applications across the Mac platform
- Learn to perform sophisticated image processing operations with CopyBits( ) and CopyDesk( )
- Learn to enable your applications to take advantage of QuickDraw compliant graphics hardware and accelerators.

List $26.95 Our price **$24.25**

Optimizing PowerPC Code: Programming the PowerPC in Assembly Language



- Take full advantage of the potential of the PowerPC by mastering the Assembly Language techniques.
- Learn to produce faster more robust software!

List $39.95 Our Price **$35.96**

Inside Macintosh®: PowerPC Numerics

by Apple Computer, Inc.

- Describes the floating-point numerics environment provided with the first release of PowerPC processor-based Macintosh computers.
- Provides a description of the IEEE standard 754 for binary floating-point arithmetic., and how RISC Numerics compiles with it.
- Shows programmers how to create floating-point values and how to perform operations on floating-point values in high-level languages such as C and in PowerPC assembly language.

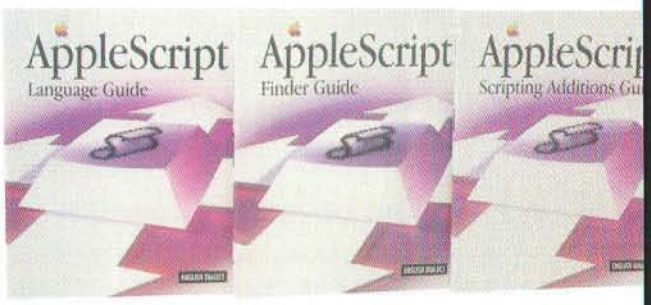
List $28.95 Our Price **$26.00**

Inside Macintosh®: PowerPC System Software by Apple Computer, Inc.

- Describes the new process execution environment and system software services provided with the first version of the system software for Macintosh on PowerPC computers.
- Contains information to write applications that can run on the PowerPC.
- Shows how to make your software compatible with the new run-time environment provided on PowerPC-based Macintosh computers. It also provides a complete technical reference for the Mixed Mode Manager, the Code Fragment Manager, and the Exception Manager.

List $24.95 Our Price **$22.45**

AppleScript Finder Guide, English Dialect by Apple Computer, Inc.



- Provides definitions for Finder object classes and commands.
- Write, record, or run scripts that trigger the same desktop actions that you trigger using the keyboard and mouse.

List $19.95 Our Price **$17.95** SEE RELATED CATEGORY: **Scripting**

AppleScript Language Guide,

by Apple Computer, Inc.

- A complete reference for anyone using AppleScript to modify existing scripts or to write new ones.
- Contains useful information for programmers who are working on scriptable applications or complex scripts.
- Features detailed definitions of AppleScript terminology and syntax in the following categories: Value classes, commands, objects and references to objects, expressions, control statements, handlers, and script objects.
- Includes many sample scripts, discusses advanced topics such as writing command handlers for script applications, the scope of script variables and properties declared at different levels in a script, and inheritance and delegation among script objects.

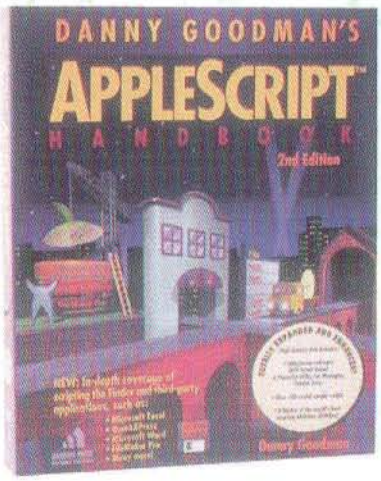
List $29.95 Our Price **$26.95** SEE RELATED CATEGORY: **Scripting**

AppleScript Scripting Additions Guide

by Apple Computer, Inc.

- Use the standard scripting additions commands.
- Write scripting additions.

List $18.95 Our Price **$17.05** SEE RELATED CATEGORY: **Scripting**



Danny Goodman's AppleScript Handbook Second Edition by Danny Goodman

- Customize and extend the capabilities of any Macintosh computer – no programming experience needed!
- Learn to use scripts to enhance the Macintosh environment, automate many processes, link data between applications, and much more.
- All-new examples showing how to integrate AppleScript with the Finder, spreadsheets, desktop publishing programs, graphics applications, databases, telecommunications programs, utilities, and HyperCard.
- Includes 3 1/2" disk with over $100 worth of software, including AppleScript 1.1, valuable utilities, and powerful, ready-to-use scripts.

Includes Disk!

List $39.00 Our Price **$35.00**

SEE RELATED CATEGORY: Scripting

SEE RELATED CATEGORY: Tools, Libs & Utilities

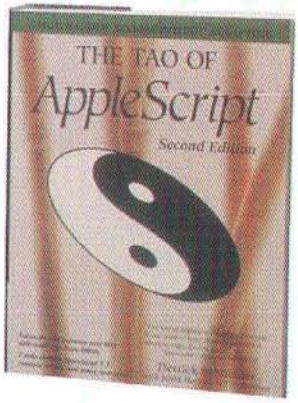
Applied Mac Scripting

SEE RELATED CATEGORY: Scripting

- Learn to design and develop powerful scripts.
- Covers AppleScript™, Frontier, QuicKeys, Tempo II, nShell, FaceSpan Application Builder, Scripting PlainTalk and System 7.5.
- Hands on tutorial shows you how to automate your Macintosh activities by learning how to use the AppleScript and Frontier scripting environments.
- Harness the capabilities of a wide variety of Macintosh applications into the integrated productivity tools. This includes such things as the newspaper script which combines the power of SITcomm, MacWrite Pro, and FileMaker Pro, or QuarkXPress.

List $34.95 Our Price **$31.45**

The Tao of AppleScript: BMUG's Guide to Macintosh Scripting, Second Edition by Derrick Schneider & Hans Hansen



- A complete, natural introduction to AppleScript programming essentials.
- Learn to customize applications, automate tedious tasks, and create programs without having to use a complex programming language.
- Includes 2 disks containing AppleScript, QuickTime, StuffIt Lite, ResMover, and other helpful utilities.
- Loaded with practical examples for easy learning

List $29.95 Our Price **$26.95**

SEE RELATED CATEGORY: Scripting

Includes Disk!

ScriptBase™ by Main Event Software

- A database for storing persistent objects to be made available for access to AppleScript, Apple's system-level user scripting language for controlling applications on Macintosh® computers.
- Once installed, the database becomes part of the AppleScript system, adding a host of commands to the basic AppleScript vocabulary. Retrieving the objects is simple using AppleScript's natural-language syntax and structure. Objects stored and retrieved in ScriptBase can be accessible any time from any script on the user's computer. These objects can be of any type, including numbers, character strings, lists, records, scripts, and references to disks, files, folders, as well as abstract raw data, to name just a few.
- ScriptBase can be used to maintain system-wide settings, such as sets of preferences, paths to frequently-used files or folders. Complex installations can be made easier by organizing data and scripts within the database's structure.

Our Price **$79.00**

SEE RELATED CATEGORY: Scripting

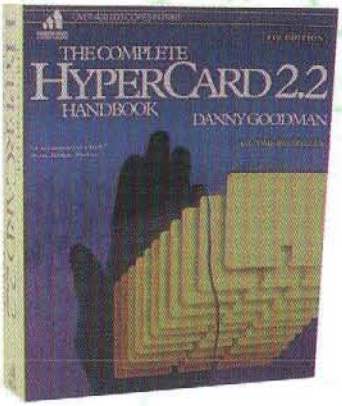
The Complete AppleScript Handbook by Danny Goodman

Master AppleScript with the definitive book/disk toolkit. This self-contained toolkit teaches you AppleScript from the ground up providing you with tools you need to automate tasks and integrate Macintosh applications. This book explains in detail all commands and usage of the AppleScript language including:

• Issuing command • Describing objects • Working with values, variables, and expressions • Using if-then constructions, loops and subroutines • Error checking and debugging • Scripting Finder-level processes • Using AppleScript with third party applications

List $35.00 Our Price **$31.50**

SEE RELATED CATEGORY: Scripting



The Complete HyperCard® 2.2 Handbook Fourth Edition

by Danny Goodman

- The biggest-selling programming Mac book.
- Learn to build working applications using the latest version of HyperCard.
- Covers text, painting tools, extension commands (XCMDs), scripting in HyperTalk, and more.

List $35.00 Our Price **$31.50**

SEE RELATED CATEGORY: **Scripting**

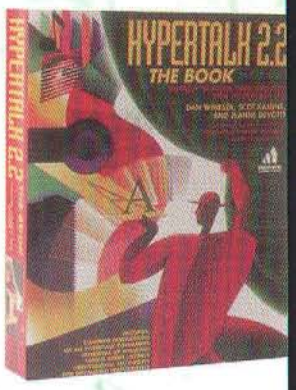
HyperTalk® 2.2: The Book Second Edition

by Dan Winkler, Scott Kamins, and Jeanne DeVoto

SEE RELATED CATEGORY: **Scripting**

- The most complete, authoritative source on HyperTalk 2.2 programming and troubleshooting.
- Covers each language element of HyperTalk 2.2 (including the odd quirk or bug).

List $35.00 Ours Price **$31.50**



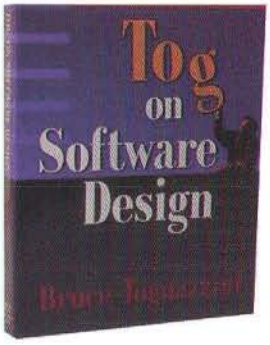
HyperCard Stack Design by Apple Computer, Inc.

HyperCard Stack Design Guidelines is an essential book for everyone who creates Apple® HyperCard stacks. Included are illustrated discussions of:

- Guidelines for stack development-audience evaluation, subject matter requirements and constraints, mode of presentation, and testing
- Navigation, the importance of making sure users can get around in your stacks
- Graphic Design and illustration- including the use of grids to determine card and background layout
- Text in stacks- placement, readability, and special considerations when writing for the screen
- Music and sound in stacks-as subject matter, reinforcement, entertainment, alert mechanism, or transition

List $21.95 Our Price **$19.95**

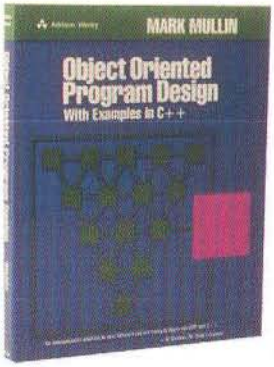
SEE RELATED CATEGORY: **Scripting**



Tog on Software Design by Bruce "Tog" Tognazzini

Respected industry futurist, Tog, presents his vision of our technological future, detailing the steps computer professionals need to take to deliver new technologies that will profit the industry and benefit society in general. Contains Tog's insights on a wide range of topics from quality management to the meaning of standards, and responses to queries supplied by designers and developers.

List $29.95 Our Price **$26.95**



Object Oriented Program Design by Mark Mullin

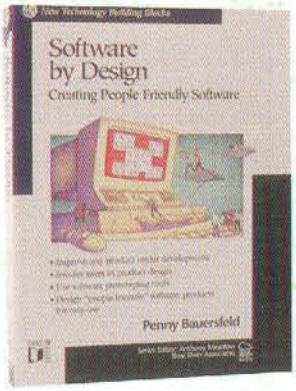
- A concise guide to the essential concepts and techniques of OOP design
- Clarifies the key concepts of object oriented programming such as objects, classes, entities, hierarchies, and inheritance
- Uses typical database application to illustrate each OOP topic, to give the programmer a familiar point of reference

List $22.95 Our Price **$20.66**

Lowest price guaranteed. Feel Good about your purchase! 30-day return policy.

Software By Design: Creating User Friendly Software

by Penny Bauersfeld (Series Editor: Tony Meadow)



- A thorough how-to for designing software that is easy to learn and comfortable to operate.
- Written from the Macintosh perspective, but compatible with all platforms.
- Stresses user input from initial design, through prototyping, testing and revision.
- Provides tools for analyzing user needs and test responses, plus exercises for sharpening user-oriented design skills.

List $29.95 Our Price **$26.95**

Inside Macintosh®: CD-ROM by Apple Computer, Inc.

- More than 25 volumes in electronic form.
- Includes: QuickDraw™ GX Library, Macintosh Human Interface Guidelines, PowerPC System Software, Macintosh Toolbox Essentials and More Macintosh Toolbox, QuickTime and QuickTime Components.
- Access over 16,000 pages of information with Hypertext linking and extensive cross referencing.

List $99.95 Our Price **$89.95**

Inside Macintosh®: Sound by Apple Computer, Inc.

- Describes the parts of the Macintosh system software that allow you to manage sounds.
- Contains information to write applications that can record and play back sounds, compress and expand audio data, convert text to speech, and perform other similar operations.

List $26.95 Our Price **$24.25**

Inside Macintosh®: Imaging by Apple Computer, Inc.

- Covers QuickDraw and Color QuickDraw.
- Includes general discussions of drawing and working with color.
- Describes the structures that hold images and image information, and the routines that manipulate them.
- Covers the Palette, Color, and Printing Managers, and the Color Picker, Color Matching, and Picture Utilities.

List$26.95 OurPrice **$24.25**

Inside Macintosh®: QuickTime by Apple Computer, Inc.

- For developers who want to create applications that use QuickTime, the system software that allows the integration of video, animation, and sounds into applications.
- Describes all of the QuickTime Toolbox utilities.
- Provides the information you need to compress and decompress images and image sequences.

List $29.95 Our Price **$26.95**

Inside Macintosh®: QuickTime Components by Apple Computer, Inc.

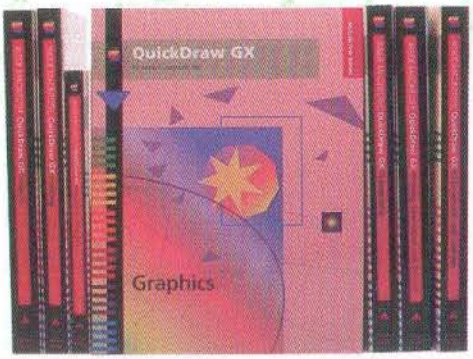
Covers how to use and develop QuickTime components such as image compressors, movie controllers, sequence grabbers, and video digitizers.

List $34.95 Our Price **$31.45**

Inside Macintosh®: QuickDraw™ GX Programmer's Overview

- Provides an introduction to QuickDraw™ GX, providing an overview of the QuickDraw GX environment from a developer's perspective.
- Introduces the QuickDraw™ GX programming and runtime environments, the relationship between QuickDraw GX and the rest of the Macintosh® systems software and the relationship between QuickDraw GX and Macintosh applications.
- Learn the key elements of QuickDraw GX programming, data structures, object types, and functions used most frequently by QuickDraw GX developers are also covered.
- Provides a series of practical examples demonstrating how to approach programming with QuickDraw GX.

List $24.95 Our Price **$22.45**



Inside Macintosh®: QuickDraw™ GX Graphics
by Apple Computer, Inc.

- Shows how to create and manipulate the fundamental geometric shapes of QuickDraw GX to generate a vast range of graphic entities.
- Demonstrates how to work with bitmaps and pictures, and specialized QuickDraw GX graphic shapes.

List $26.95 Our Price **$24.25**

Inside Macintosh®: QuickDraw™ GX Objects
by Apple Computer, Inc.

Introduces QuickDraw GX and its object structure, and shows programmers how to manipulate objects in all types of programs.

List $26.95 Our Price **$24.25**

Inside Macintosh®: QuickDraw™ GX Printing by Apple Computer, Inc.

- Essential for any developer whose QuickDraw™ GX application supports printing.
- Shows how to support the new printing features of QuickDraw GX, including desktop printers and expandable printing dialog boxes.
- Shows how to use printing-related objects to add custom panels to printing dialog boxes and to create custom page formats.

List $26.95 Our Price **$24.25**

Inside Macintosh®: QuickDraw™ GX Printing Extensions and Drivers by Apple Computer, Inc.

- Essential for developers who want to create extensions to the application printing capabilities of QuickDraw™ GX, or who need to write a printing device driver that works with QuickDraw GX.
- Describes how to create printing extensions and printer drivers, and provides a complete reference to the messages, functions, and resources that they use.

List $29.95 Our Price **$26.95**

Inside Macintosh®: QuickDraw™ GX Typography by Apple Computer, Inc.

- Essential for developers who use QuickDraw™ GX to manipulate text.
- Shows how to use QuickDraw GX objects to handle all kinds of text – from plain, unstyled text to complex, mixed-direction and multi-language text with sophisticated stylistic and typographic variations.
- Shows how to create and manipulate the three different types of text shapes supported by QuickDraw GX including text shapes, glyph shapes, and layout shapes.

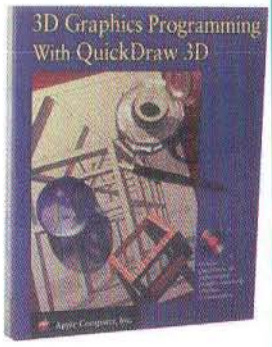
List $29.95 Our Price **$26.95**

Inside Macintosh®: QuickDraw™ GX Environment and Utilities by Apple Computer, Inc.

- Companion to QuickDraw™ GX Objects.
- Contains programming information useful to any developer writing QuickDraw GX applications.
- Describes QuickDraw GX memory management, error handling, debugging, and mathematical functions, as well as conversion from QuickDraw to QuickDraw GX.

List $29.95 Our Price **$26.95**

3D Graphics Programming Using QuickDraw 3D
by Apple Computer, Inc.



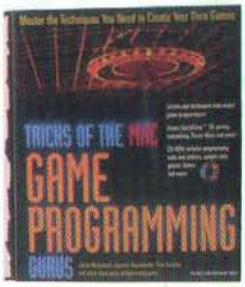
- Incorporate spectacular 3D graphics into your applications.
- Explore QuickDraw 3D, a revolutionary graphics extension to the Mac OS for Power Macintoshes.
- CD contains the complete QuickDraw 3D system itself and a complete database of the QuickDraw 3D API, allowing you instant access to the hundreds of graphics calls via a fast viewing engine. Book/CD-ROM, 640 pages.

List $39.95 Our Price **$35.96**

3D Game Machine v1.2 by Virtually Unlimited

- Create lightning-fast 3D arcade games and interactive multimedia applications.
- Ultra-fast rendering – 15 frames per second on a 14" monitor completely texture-mapped, with a PowerMac 6100/60
- Create full "virtual" 3D worlds with six degrees of freedom, free-form texture mapping, shading, material and light properties, convex/cave polygons with unlimited vertices, unlimited light sources, dynamic hidden surface removal, special graphic modes for fast full-screen animation, collision detection, explosion simulation, 3D data importing.
- Simple easy-to-use interface. Runs on all Macs! Works with CodeWarrior.

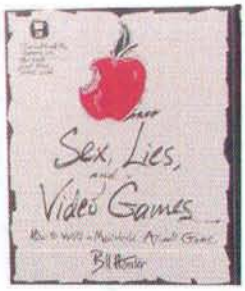
Our Price **$299.00** + license.



Tricks of The Mac Game Programming Gurus

- For beginning to expert game programmers
- Complete overview of all the necessary components of game programming on the Macintosh.
- Packed with valuable tools, utilities, sample code, CodeWarrior™ Lite and game demos.
- QuickDraw 3D and Power Mac optimization and inside info on how Glypha III was created.
- Hundreds of tried-and-true tricks, tips, and insider secrets from well-known Mac game programming experts

List $50.00 Our Price **$45.00**



Sex, Lies and Video Games by Bill Hensler

- A learn-by-example tutorial on the ins and outs of Mac arcade-style game programming in C.
- Features game theory, sprite animation, sound, and interaction techniques.
- A must-read for serious programmer's and hobbyists alike.

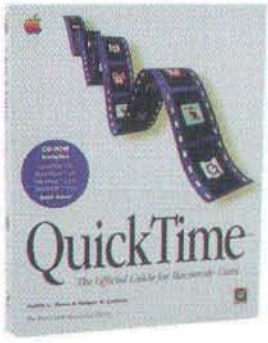
List 34.95 Our Price **$31.46**

QuickTime Official Guide 4 Mac

SEE RELATED CATEGORY:
Tools, Libs & Utilities

- Includes step by step lessons and it's hands on advice help you to get the most out of QuickTime
- Customize the way you play QuickTime movies
- Digitize your own video
- Cut and Paste clips and add tracks to movies
- With CD that includes QuickTime 2.0, VideoShop 2.0.3, QuickFLIX 1.1.1, and MoviePlayer 2.0
- Incorporate sound, music, and MIDI support in your movies
- Edited by Don Crabb

List $49.50 Our price **$45.00**



QuickTime Starter Kit For The Macintosh by Robert A. Lettieri & Judith Stern

- The ultimate package for getting productive and having fun with Macintosh movie-making
- Play, make, and edit QuickTime movies easily
- Loaded with tips on the best ways to bring live-action video to Mac multimedia
- CD-ROM includes QuickTime tools, movie clips, shareware, and demos of Premiere and other programs.

List $45.00 Our Price **$40.50**

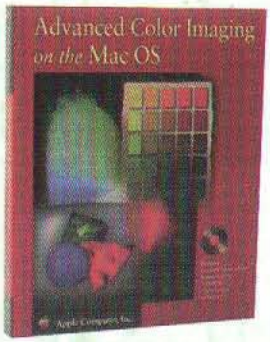
SEE RELATED CATEGORY:
Tools, Libs & Utilities

Graphic Gems V Edited by Alan W. Paeth

- Loaded with practical tools for implementing new ideas and techniques, to offer working solutions to real programming problems.
- Written by a diverse group of graphics programmers from industry, academia, and research.
- Contains over 40 new gems in ellipses, splines, Bezier curves, and ray tracing – displaying the most recent and innovative techniques in graphics programming.
- Includes a disk with source code from all five volumes. Available in both IBM and Macintosh versions. CONTENTS: Algebra and Arithmetic. Computational Geometry. Modeling and Transformation. Curves and Surfaces. Ray Tracing and Radiosity. Halftoning and Image Processing. Utilities.

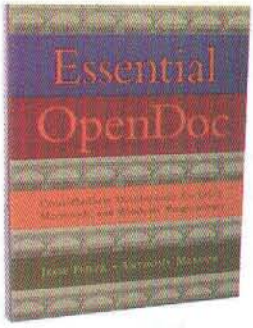
List $49.95 Our Price **$44.95**

Advanced Color Imaging on the Mac OS



- Enhance your software's color capabilities with step-by-step instructions.
- Augment the color support supplied with QuickDraw, and QuickDraw GX.
- Use the Pallette Manager to get the best colors on limited displays.
- Match colors between screens and input/output devices (scanners & printers)
- CD includes a complete reference information in both QuickView and Acrobat formats. Plus, a sample application demonstrating ColorSync programming techniques.

List $36.95 Our Price **$33.25**



OpenDoc Programmer's Cookbook

- Shows you how to create OpenDoc software components, called parts editors, for the Mac OS Platform.
- Including instructions for setting up the Macintosh Programmers Workshop (MPW) development environment to write OpenDoc software
- Annotated listings of explaining the methods that implement the SamplePart part editor
- Descriptions of other sample part editors created by the OpenDoc engineering team to illustrate more advanced features
- Summary descriptions of software utilities provided with OpenDoc for the Mac OS
- An Introduction to the System Object Model (SOM) technology underlying OpenDoc

List $24.95 Our price **$22.45**

Essential OpenDoc

- Gives an in-depth look at the technical issues of OpenDoc
- Explores the three core technologies that support it's functionality – SOM, OpenDoc's storage mechanism, and the Open Scripting Architecture (OSA).
- Also examines CyberDog, a set of OpenDoc part editors that provides access to Internet services and offers compelling example of the power of OpenDoc development

List $39.95 Our price **$35.95**

Inside CodeWarrior 9

- Includes CodeWarrior IDE User's Guide.
- This is the printed version of the documentation provided on the CD.
- Covers CodeWarrior, the debugger, and associated tools.

Our Price **$34.95**

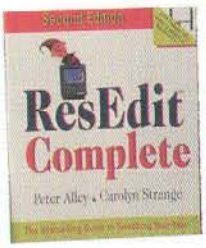
SEE RELATED CATEGORY:
Dev. Environments

ResEdit™ Complete, Second Edition

by Peter Alley and Carolyn Strange

- Customize every aspect of your interface form creating screen backgrounds and icons to customizing menus and dialog boxes. 608 pages. Book/disk package.

List $34.95 Our Price **$31.45**



Inside PowerPlant Manual

- Create PowerPlant applications using the CodeWarrior IDE and PowerPlant Constructor.
- Full descriptions of major PowerPlant classes and resources.
- Included are the PowerPlant Constructor Manual, including View, TextTraits and Custom Types editing, and PowerPlant Library Reference, covering all classes and functions in PowerPlant.

Our Price **$34.95**

SEE RELATED CATEGORY:
Dev. Environments

OpenDoc Programmer's Guide by Apple Computer, Inc.

- The official reference for the implementation of OpenDoc on the Mac OS.
- Describes the component software revolution and explains how to develop for it on the Mac OS platform.
- Accompanying CD-ROM contains a complete reference to the OpenDoc programming interface, and an extensive collection of tested, reusable sample code.

List $44.95 Our Price **$40.46**

One & 2-Day SHIPPING (call for details)

C++ Programming with CodeWarrior by Jan L. Harrington

- Beginning OOP for the Macintosh and Power Macintosh and Mac OS compatibles.
- Learn object-oriented programming techniques using C++ as the example language and Metrowerks and CodeWarrior as the example compiler.
- Enclosed CD contains example code from the book and a full-function Metrowerks CodeWarrior

List $35.95 Our Price **$32.35**

The ResEdit All Night Diner by David Ciskowski

- An idea-filled menu and introduction to the joys of customizing software.
- Add personality to the Mac by customizing default icons, the text of menus and dialog boxes, cursors, pointers and more.
- Disk features ResEdit, plus lots of sample resources

List $24.95 Our Price **$22.45**

SEE RELATED CATEGORY: **Dev. Environments**

C++ Programming W/MacApp by David Wilson, Larry Rosenstein & Dan Shafer

- Learn the secrets to unlocking the power of MacApp®, Apple's development environment for C++
- Learn to design complex windows and views using the ViewEdit tool
- Learn to support multipage text and graphics with only five lines of code
- Learn to support Undo for menu commands and drawing operations that use the mouse.

List $34.95 Our Price **$31.46**

SEE RELATED CATEGORY: **Tools, Libs & Utilities**

Programming in Symantec C++ for the Macintosh by Judy May and John Whittle

- An introduction to object-oriented programming, the C++ language, and Symantec C++ for the Macintosh.
- Great for both programmers and beginners alike.
- Covers everything from the basics to advanced features of Symantec C++.
- Includes helpful examples of C++ code that illustrate object-oriented programs.

List $29.95 Our Price **$26.95**

SEE RELATED CATEGORY: **Dev. Environments**

Symantec C++ Programming by Neil Rhodes & Julie McKeehan

Symantec C++ Programming for the Macintosh is a tutorial for getting up and running in the Symantec C++ environment, while mastering the techniques of object-oriented programming.

- Explore the Symantec C++ environment, from debugging a program and using resource utilities to building applications and creating objects
- Design programs for compatibility with multiple application frameworks
- Learn how to use the new Visual Architect, Inspector, and templates.

List $45.00 Our Price **$40.50**

SEE RELATED CATEGORY: **Dev. Environments**

Metrowerks CodeWarrior Programming by Dan Parks Sydow

- Includes CodeWarrior Lite, and Full Coverage of PowerPlant™.
- The best information on Metrowerks CodeWarrior, giving full coverage to the Gold Edition.
- CD includes Code Warrior Lite.

List $39.95 Our Price **$35.95**

SEE RELATED CATEGORY: **Dev. Environments**

Dan Shafer Presents the Power of Prograph CPX

- Master the revolutionary graphical object-oriented programming language.
- Step by step course through three interrelated projects of increasing complexity.
- Learn Prograph language, CPX classes and object editors.
- Includes disk with all code in the book.

List $49.95 Our Price **$44.95**

Order Toll-free
800-MACDEV-1
(800-622-3381)

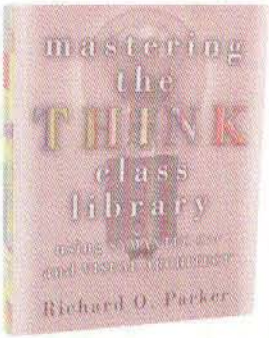
Visual Programming with Prograph CPX

by Scott B. Steinman and Kevin G. Carver

- An introduction to the language and a guide for advanced users, for both Macintosh and Windows-based machines.

List $34.00 Our Price **$30.60**

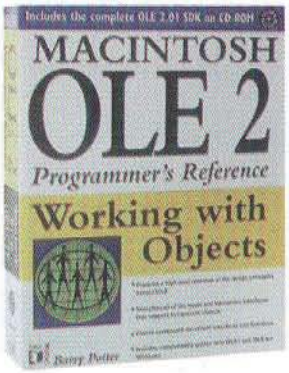
Mastering the THINK Class Library by Richard Parker



- Provides a thorough examination of Symantec's extensive Class Library and the Visual Architect.
- A complete description of the structure and operation of the TCL includes explanations of all code generated by the Visual Architect, any necessary custom code, and the operation of this code.
- Visual Architect tutorials provide you with a step-by-step approach for simplifying the development of complex Macintosh applications. 496 pages.

List $29.95 Our Price **$26.95**

SEE RELATED CATEGORY: **Dev. Environments**



Macintosh OLE2 Programmer's Reference: Working with Objects

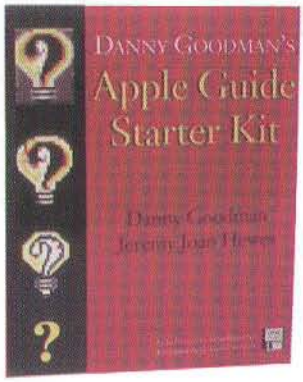
- Complete reference to the extensible protocol of Object Linking and Embedding, version 2.01 for Macintosh System 7.
- Describes the visual and interactice interfaces that support the component objects.
- Provides details of the OLE 2.01 for the Macintosh user Interface, addresses the issues of object class registration, shows how to implement the drag and drop objects from one application to another, covers the interface that exposes the basic embedding functionality, includes descriptions of API functions and more!

List $44.95 Our Price **$40.45**

Includes CD-ROM!

Danny Goodman's Apple Guide Starter Kit

by Danny Goodman and Jeremy Joan Hewes.



- Create your own Apple Guide databases quickly and easily, without having to learn a scripting language, write coded files, or use several different files and programs
- Includes advice and tips on how to design a good Guide, from planning and creation through testing, revising, and indexing. Book/disk, 320 pages.

List $34.95 Our Price **$31.46**

MacsBug Reference & Debugging Guide For MacsBug version 6.2

by Apple Computer, Inc.

- MacsBug is an assembly-language-level debugging tool
- Macros, templates, dcmds, and other resources for making debugging easier
- Macintosh memory management and the operating system as they relate to low level debugging
- How to display and set memory and process registers
- Disassemble memory, and set execution breakpoints
- Discipline, a tool for testing the validity of toolbox parameters
- Debugging strategies you can use to find and cure common bugs
- Includes MacsBug 6.2.

List $34.95 Our Price **$31.46**

AppleGuide Complete

by Apple Computer, Inc.

- Covers Guide Maker, the software you use to build and test guide files.
- Learn about the complete cycle of designing as well as advanced topics such as scripting and coding guide files. Book/CD-ROM, 544 pages.

List $39.95 Our Price **$35.96**

Includes CD-ROM!

SEE RELATED CATEGORY: **Tools, Libs & Utilities**

Programming For The Newton: Software Development using NewtonScript

by Julie McKeehan and Neil Rhodes. Foreword by Walter R. Smith

- An indispensable tool for Newton programmers.
- Includes disk with sample Newton application from the books, as well as demonstration version of Newton Toolkit (NTK) – the complete development environment for the Newton®.
- A Publication of AP Professional May 1994, Paperback, 393 pp.

List $29.95 Our Price **$26.95**

SEE RELATED CATEGORY: **Dev. Environments**

Lowest price guaranteed. Feel Good about your purchase! 30-day return policy.

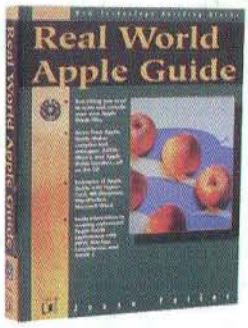
Real World AppleGuide For The Mac

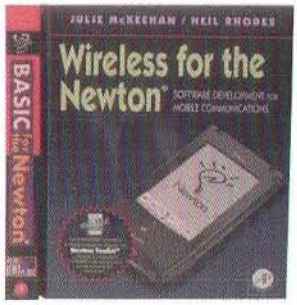
SEE RELATED CATEGORY: Tools, Libs & Utilities

- An introduction to AppleGuide for programmers. It explains its design and function, plus how to design your own guides using AppleScript.
- Includes a disk of sample AppleGuides for AppleGuide-compliant applications.

Includes Disk!

List $39.95 Our Price **$35.95**





Wireless For The Newton: Software Development for Mobile Communications

by Julie McKeehan and Neil Rhodes

- Learn to develop Newton® software on the Macintosh.
- Hands-on Newton environment training with sample code
- Includes disk with sample source code for a Newton application, as well as demonstration NTK™ – the complete development environment for the Newton®.

List $34.95 Our Price **$31.45**

Programming for the Newton Using NS BASIC

by John Schettino & Liz O'Hara

Includes Disk!

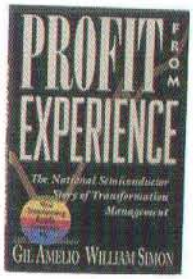
- Program on Macintosh, Windows-based PC, or on the Newton itself.
- Straight-forward "programming by example" approach – you'll be writing Newton programs right away.
- Includes 3.5" disk containing Demonstration NS BASIC and over fifty example programs. (Newton not included)

List $35.95 Our Price **$32.35**

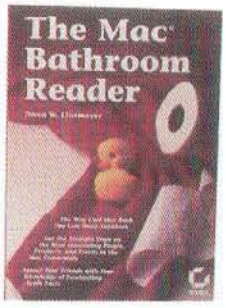
SEE RELATED CATEGORY: Dev. Environments

Profit From Experience by Gil Amelio and William Simon

Written by the new CEO of Apple



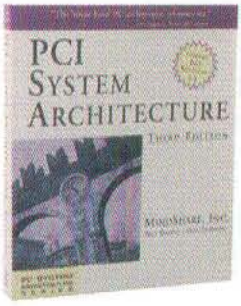
- The story of the transformation of National Semiconductor – how Amelio and his management team took it from it's worst loss in 30 years to the highest earnings in it's history • Includes: 6 core business issues and why they are critical to success • The TEAM program • 10 personal attributes to strive for • Amelio's guidelines for General Managers • Attributes to look for when hiring. List $24.99 Our price **$22.45**



The Mac Bathroom Reader by Owen Linzmayer

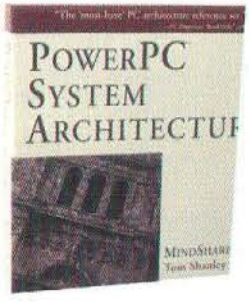
- Get the straight dope on the most interesting people, products, and the events in the Mac community
- Amaze your friends with your knowledge of fascinating Apple facts
- Won't improve your Mac skills, but it will will help you impress your friends with useless Mac trivia.

List $12.99 Our Price **$11.70**



PCI System Architecture, Third Edition by MindShare

Describing revision 2.1 of the Peripheral Component Interconnect (PCI) bus specification, this book explores PCI's relationship to the rest of the system. It includes an in-depth treatment of PCI to PCI bridges, the PCI BIOS, the 66MHz PCI bus, and more. 592 pages. List $34.95 Our Price **$31.46**

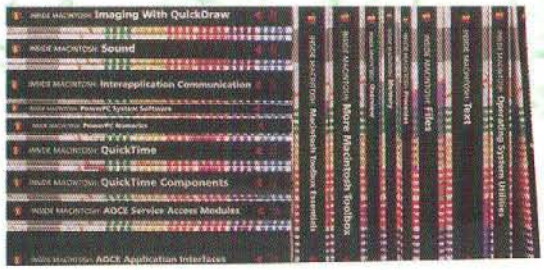


PowerPC System Architecture

by MindShare Inc. and Tom Shanley

- Describes the hardware architecture of the PowerPC systems, providing clear, concise explanation of the PowerPC specification, the template upon which all PowerPC processors are designed.
- Includes the specs for both the 32 and 64 bit implementations including • supervisor privilege level facilities • logical memory addresses • I/O and memory mapped I/O • address translation for segments, pages, and blocks • virtual paging • interrupts.

List $34.95 Our Price **$31.46**



Inside Macintosh®: AOCE Application Interfaces

by Apple Computer, Inc.

- Shows how your application can take advantage of the system software features provided by PowerTalk system software and the PowerShare collaboration servers.
- Add electronic mail capabilities to your application, write a messaging application or agent, store information in and retrieve information from PowerShare and other AOCE catalogs.
- Add catalog-browsing and find-in-catalog capabilities to your application, write templates that extend the Finder's ability to display information in PowerShare and other AOCE catalogs.
- Add digital signatures to files or to any portion of a document, and establish an authenticated messaging connection.

List $40.45 Our Price **$36.40**

Inside Macintosh®: Interapplication Communication

by Apple Computer, Inc.

Shows how applications can work together. How your application can share data, request information or services, allow the user to automate tasks, communicate with remote databases.

List $34.95 Our Price **$31.45**

Inside the Macintosh Communications Toolbox

by Apple Computer, Inc.

- The definitive reference to the Macintosh Communications Toolbox, an integral part of the System 7 Macintosh Toolbox that enables developers to create communications applications or add communications features to other applications.
- Describes all of the routines that provide programmers with standard access to important communications services and in addition enables programmers to extend the reach of the Macintosh into non-Apple environments.

List $24.95 Our Price **$22.95**

Inside Macintosh®: Networking

by Apple Computer, Inc.

- Describes how to write software that uses AppleTalk networking protocols.
- Describes the components and organization of AppleTalk and how to select an AppleTalk protocol.
- Provides the complete application interfaces to all AppleTalk protocols, including ATP (AppleTalk Transaction Protocol), DDP (Datagram Delivery Protocol), and ADSP (AppleTalk Data Stream Protocol), among others.

List $29.95 Our Price **$26.95**

Inside Macintosh®: AOCE Service Access Modules by Apple Computer, Inc.

- Describes how to write a software module that gives users and PowerTalk-enabled applications access to a new or existing mail and messaging service or catalog service.
- Shows how to write a catalog service access module (CSAM), a messaging service access module (MSAM), and AOCE templates that allow a user to set up a CSAM or MSAM and add addresses to mail and messages.

List $26.95 Our Price **$24.25**

Inside Macintosh®: Text by Apple Computer, Inc.

- Describes how to perform text handling, from simple character display to multi-language processing.
- Covers Font, Script, Text Services, and Dictionary Managers, in addition to QuickDraw Text, TextEdit, and International and Keyboard Resources.

List $39.95 Our Price **$35.95**

Inside Macintosh®: Devices by Apple Computer, Inc.

- Describes how to write software that interacts with built-in and peripheral hardware devices.
- Learn how to write and install your own device drivers, desk accessories, and Chooser extensions.
- Communicate with device drivers using the Device Manager; access expansion cards using the Slot Manager; control SCSI devices using SCSI Manager 4.3 or the original SCSI Manager.
- Communicate directly with Apple Desktop Bus devices; interact with the Power Manager in battery-powered Macintosh computers, and communicate with serial devices using the Serial Driver.

List $29.95 Our Price **$26.95**

Inside Macintosh®: X-Ref by Apple Computer, Inc.

Is a fast access to all the information in Inside Macintosh. Inside Macintosh X Ref; Provides programmers with a quick and easy way to find the exact information they need in this definitive suite of books, (all 26 volumes). It is indexed by topic, volume, chapter, and accompanying page number.

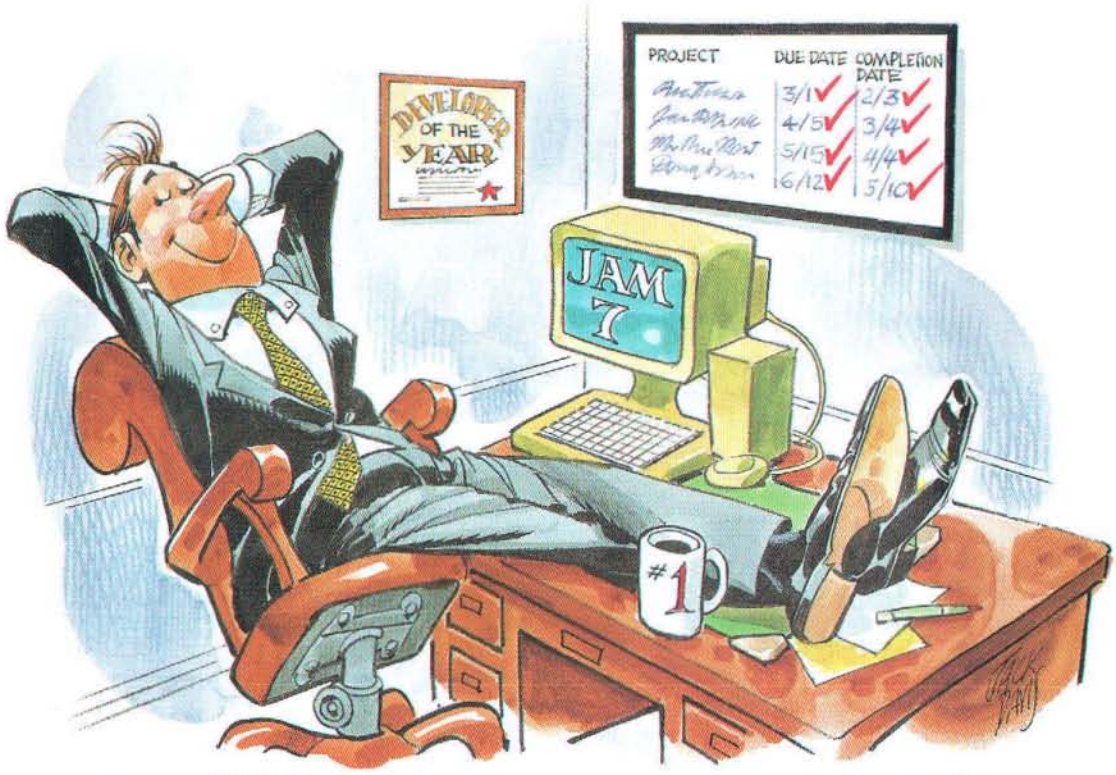
List $19.95 Our Price **$17.95**

Order Toll-free
800-MACDEV-1
(800-622-3381)

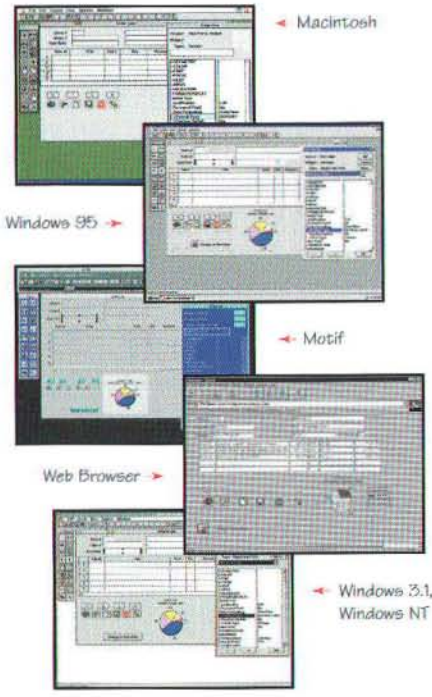
All entries in this index are alphabetized.
For your convenience the product names are **bold**, book names are *italicized* and company names are in plain text.

INDEX

Life is Sweet with JAM!



Get JAM and enjoy success in building and deploying client/server and Web applications



In a world where requirements change and deadlines don't, only JAM gives you what you need to build powerful, completely portable client/server, 3-tier and Web applications. JAM's cross-platform RAD capabilities enable you to successfully develop and deploy demanding applications across any platform, database, GUI or Web Browser.

Are you willing to base your project's success on a tool that only *promises* portability? JAM lets you build great looking applications that are fully portable *today*. If you need to build serious client/server, 3-tier or Web applications, call JYACC for a free demonstration kit or white paper.

- ✓ Desktop ease with the power of a 2nd-generation tool
- ✓ Unrivaled portability between Windows® 3.1, NT, 95; Macintosh®; OS/2 Warp®; Character, Motif and Web Browsers
- ✓ Unparalleled database access to Oracle®, Sybase®, Informix®, ODBC and more
- ✓ Automatic SQL generation with full transaction control
- ✓ Visual repository-driven development of both client and application server
- ✓ Centralized control over application objects via inheritance
- ✓ Unique architecture for team development
- ✓ No runtimes

Find out for yourself how sweet life can be with JAM!

Call 1 800 458-3313.

Or E-mail: sweetlife3@jyacc.com for a free demonstration kit or white paper.
Visit our Web site at http://www.jyacc.com
For international inquiries call: 1-212-267-7722 or FAX 1-212-608-6753

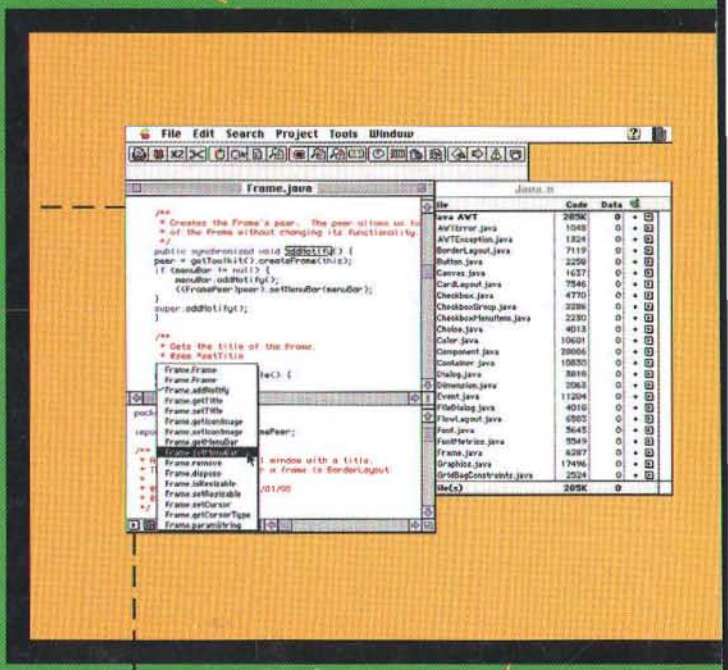
JYACC
Meeting the needs of professional developers for over 18 years

BRAZIL (55) 11 816 6228 • FINLAND (358) 3486 4000 • FRANCE (33) 1 46 92 45 44 • GERMANY (49) 40 79 70 07 0 • HRVATSKA (385) 1 242 116 • ISRAEL (972) 3 557 3675
ITALY (39) 2 45 2701 • MEXICO (52) 5 663 0405 • RUSSIA (7) 095 288 1924 • SAUDI ARABIA (966) 2 665 8406 • SINGAPORE (65) 220 8322 • SLOVENIA (386) 61 1405 004
SPAIN (34) 1 804 0625 • SWEDEN (46) 8 15 10 10 • SWITZERLAND (41) 21 991 9041 • THAILAND (66) 2 513 3559 • THE NETHERLANDS (31) 70 320 9214 • UNITED KINGDOM (44) 171 814 6660

The future is definitely not running away from you

Java™ tools for CodeWarrior 9 premiering May 13 in San Jose, CA.

CodeWarrior, the award-winning Mac-hosted development suite now includes full-featured Java™ tools! CodeWarrior 9 introduces the Java compiler, linker, preference plug-ins, source-level debugging, and integrated Java class browsing! Plus a full-featured Java development environment that you could only expect from CodeWarrior.



CodeWarrior®

CodeWarrior Gold 9

The choice for multiple OS development from the Mac™ OS. Includes C / C++, Object Pascal and new Java compiler plug-ins for multiple targets. Compiles for 68K, Power Macintosh®, Windows® 95, Windows NT™, Be™ OS and Magic Cap™.

Discover Programming™ with Java

Learn how to program using Java. Designed for both novice and experienced programmers, Discover Programming with Java takes you step-by-step through the learning process.

For More Information Call 1-800-377-5416

System requirements: Macintosh 68020, 68030 or 68040, or Power Macintosh 601, 603 or 604 processor. Minimum of 8MB RAM, CD-ROM drive to install software.

metrowerks®